University of Zurich[UZH]

Multi-level Modelling for Upstream Text Processing

Thesis
presented to the Faculty of Arts and Social Sciences
of the University of Zurich
for the degree of Doctor of Philosophy

by

Tatyana Ruzsics

Supervisory Committee: Dr. Tanja Samardžić (principal supervisor)
Prof. Dr. Rico Sennrich

Accepted in the spring semester 2021
on the recommendation of the doctoral committee composed of
Prof. Dr. Rico Sennrich
Prof. Dr. Jan Šnajder

Zurich, 2021

«Наука есть ясное познание истины, просвещение разума, непорочное увеселение жизни, похвала юности, старости подпора, строительница градов, полков, крепость успеха в несчастии, в счастии украшение, везде верный и безотлучный спутник.»

М. В. Ломоносов

"*Science is a clear learning of the verity, the enlightenment of the mind, the pure amusement of life, the praise of youth, the support of old age, the builder of cities, regiments, the fortress of success in misfortune, the ornament in happiness, everywhere faithful and inseparable companion.*"

M.V. Lomonosov

Abstract

The long-tail nature of human language — implying that most words are infrequent – has been long recognized. It causes difficulties in processing new word forms when building NLP applications. This problem of data sparsity is often addressed with upstream text processing. The upstream NLP goal is to represent the text in a form that enables the downstream applications to learn grammatical structure more efficiently and generalize well to unseen words. In this thesis, we consider three upstream methods: subword segmentation, writing normalization and morphological inflection generation.

One way to address data sparsity problem is to tokenize text into subwords and use them as atomic units of text. Such preprocessing method is modelled as a task of *subword segmentation*. Given an input word string, subword segmentation breaks it up into constituent substrings. In this work, we focus on subwords segmentation methods, where segments are linguistically motivated and identify morphemes. For example, such method would segment the word *exceptional* as *except|ion|al*.

Sparse data becomes even a more significant problem when there is no standardized writing. As a result, there is a high degree of variation in writing the same word. For example, the Swiss German word *viel* ('much') can appear as *viel*, *viil*, *vill*, *viu*, and many other potential variations. This factor makes it challenging to develop NLP tools for processing spoken dialects or historical texts. In order to reduce data sparsity in such cases, *text normalization* is usually applied. This method aims to map all variants of the same word to one form in a related language with a standardized orthography.

As NLP keeps expanding its frontiers to encompass more and more languages, data sparsity becomes a more pressing issue. In the case of low-resourced languages, downstream applications can often benefit from integrating more explicit modelling of language's linguistic structure. One way to model it is to learn inflection generation patterns of language. We consider this method formulated as *morphological inflection generation* task. This task aims to learn a mapping from a lemma to its target inflected form, given a sequence of target inflection tags. For example, given a lemma *chiudere* 'to close' in Italian, its inflected form for the third person present indefinite tense should be generated as *chiudono* '(they) close'.

In this thesis we develop models for these three upstream NLP tasks with the goals of a) improving their performance over the previous solutions; b) making models more interpretable by aligning its decision process more closely to human intuition; c) proposing solutions which are portable between tasks and languages.

We meet our objectives with a linguistically motivated multi-level modelling. All our solutions are hierarchical: we take a character-level encoder-decoder model (cED) as a starting point and propose multi-level modifications to this system for each task. Our modifications integrate a signal extracted from the structural levels of text organization higher than characters. We consider two types of information which

can be extracted from higher-level units of text. On the one hand, we extract and integrate a subword signal from character sequences where subwords can range from individual characters to word segments and include individual words. On the other hand, we incorporate context signal, either in the form of linguistic annotation or extracted from raw sequences of words.

Our first methodological innovation is a synchronized decoding algorithm. It allows integration of a higher-level language model into the decoding stage of cED model. Such language model is trained over subwords on the target side of parallel data. We show that the resulting multi-level model improves over previous solutions on the canonical morphological segmentation task across three languages. Our approach is shown to be particularly useful for languages with concatenative morphology and regular segmentation patterns while eliminating a need for extra resources other than small annotated corpus. Further improvements on languages with less regular concatenative patterns are achieved by employing more heterogeneous resources that can be easily integrated into our multi-level approach.

Benefits of our synchronized decoding method expand beyond the task of subword segmentation. The method can be portable to any other upstream task where segmentation arises on the data's target side. In our second study, we demonstrate such portability for the task of writing normalization. To solve this task, we propose a novel multi-level approach which integrates two types of higher-level information into cED system: a word-level signal on the target side of parallel data and a context signal on the source side. We approach the former by adapting our synchronized decoding method. We propose and test several solutions regarding context integration, including hierarchical language modelling on the encoder side. Our extensive analysis of the proposed model on the task of Swiss German shows: a) all our proposed models improve over the previous solutions; b) complementarity of our modifications for two types of the higher-level signal; c) most suitable settings for the context integration given the morphological properties of Swiss German.

We continue the neural model analysis topic by developing a more sophisticated interpretability method in our third study. We propose a novel methodology for studying the knowledge of infection morphology encoded in neural models which allows scaling model analysis up. We provide a linguistic argumentation why interpreting the cED model's character-level decisions for inflection generation is outside of human intuition, and how subword-level information can help achieve higher interpretability degree. To this end, we propose a multi-level model for inflection generation and an interpretation method to analyze what such model learns. Our model combines the cross-attention component of the cED system with a static self-attention component over subwords on the input side. The interpretation method allows extraction of linguistic rules for inflection from the two attention components. We show that our multi-level model performs better or at par with the previous solutions while using less trained parameters. We find that a) our pattern extraction method applied to encoder-decoder attention weights uncovers variation in form inflection morphemes;

b) pattern extraction from self-attention shows triggers for such variation; c) both types of patterns are closely aligned with grammar inflection classes and class assignment criteria, for all three languages.

Our multi-level solutions for improving upstream processing can support the development of downstream applications and provide new material for aiding fundamental linguistic research. The methods presented in this dissertation were originally published in Ruzsics and Samardžić (2017), Ruzsics et al. (2019), and Ruzsics et al. (2021). The thesis gives a more thorough exposition of the methods, provides more background information on upstream processing, presents more experimental results and directly compares the methods to each other.

Acknowledgements

This thesis would not have been possible without many colleagues and friends.

First and foremost, I would like to thank my supervisor Tanja Samardžić for giving me this unique opportunity to research, to work on challenging problems and to push my limits to heights I was never aware before. I'm grateful for your valuable guidance, for sharing your knowledge and for supporting me throughout this project.

I would like to thank Rico Sennrich for co-supervision of this thesis: although our collaboration was short, your feedback and support helped a lot to steer my work in the last year of my PhD.

Through my work on this dissertation, I've had the pleasure to work with many great teams. I'm very grateful to Massimo Lusetti, Anne Göhring, Elizabeth Stark and Simone Ueberwasser for our joint work on normalizing Swiss German messages. I'm in debt to my colleagues of the Text Group, Olga Sozinova and Ximena Gutierrez-Vasques, for our great work on interpretability in neural networks. Thank you for listening carefully to my ideas, sharing with me some challenging deadlines and giving your feedback to numerous presentations. I'm also thankful to Chris Bentz for our collaboration on morphological complexity during my early years of PhD. I thank Peter Makarov and Simon Clematide for the inspiring and driving times during our participation in the shared task on morphological inflection. Thanks to all my colleagues from Computational Linguistics Department and URPP Language and Space. In particular, I'm grateful to those who participate in our Deep Learning reading group, for a valuable exchange during this time. I'm also thankful for our collaboration with Tannon Kew and Larissa Schmidt on Swiss German segmentation.

I thank Jan Šnajder and Rico Sennrich for kindly agreeing to be in my defence committee as well as Noah Bubenhofer and Ximena Gutierrez-Vasques for the moderation of my defence. And I thank Martin Volk and Balthasar Bickel for making this PhD project possible in the first place.

And last but not least, I would like to express my gratitude to my family and friends who supported me along my PhD path. It would not have been possible to become so efficient and organized without Sophie and Charlotte. Finally, I thank Karl for supporting me and sharing every part of this path: every challenge, every deadline and every triumph.

Contents

List of Figures

List of Tables

List of Abbreviations

NLP	**N**atural **L**anguage **P**rocessing
ED	**E**ncoder-**D**ecoder model
cED	character-level **E**ncoder-**D**ecoder model
MT	**M**achine **T**ranslation
SMT	**S**tatistical **M**achine **T**ranslation
NMT	**N**eural **M**achine **T**ranslation
OOV	**O**ut-of-vocabulary
LM	**L**anguage **M**odel
HLLM	**H**igher-level **L**anguage **M**odel
RNN	**R**ecurrent **N**eural **N**etwork
HLRNN	**H**igher-level **R**ecurrent **N**eural **N**etwork
MLP	**M**ulti-layer **P**erceptron
IGT	**I**nterlinear **G**lossed **T**ext
IMP	**I**mperative **M**ood
2sS	**S**econd person **S**ingular **S**ubject

To Karl, Sophie and Charlotte

Chapter 1

Introduction

1.1 Motivation for Upstream Processing

When humans use language to convey information, it is achieved by encoding information at multiple levels. In writing, characters can be considered to be the basic units of information. Character sequences form minimal meaning-form units, referred to as morphemes, which in turn combine into words. Finally, the sequences of words forming utterances or phrases are then used to transmit a message.

Cross-linguistically languages differ already at the level of the basic units used in writing. The most population of the world uses languages where basic units are letters of the Latin alphabet. Many languages of Eastern Europe and Asia use Cyrillic alphabet. In both writing scripts, there are separate symbols for consonants and vowels. This is not the case for many other writing systems. For example, the second most used script in the world is Chinese. It belongs to a logographic type where a symbol, called logogram represents each morpheme or word. The main difference between logograms and other writing systems is that the symbols are not linked directly to their pronunciation. Yet a different example is Arabic script where symbols are used for consonants only. In some languages which use Arabic script, vowels are optionally written with diacritics.

The cross-linguistic variation continues to grow when it comes to the strategies for composing words and combining them into phrases. In order to form words and sentences, languages use a wide variety of morpho-syntactic processes. In the following example, we illustrate strategies to mark plural on nouns across four languages:

(1) Cross-linguistic strategies for Plural Marking

a. Suffixation: Turkish

Singular		Plural	
ev	'house'	*ev-ler*	'houses'

b. Prefixation: Swahili

Singular		Plural	
m-toto	'child'	*wa-toto*	'children'

c. Reduplication: Malay

Singular		Plural	
anak	'child'	*anak-anak*	'children'

d. Base modification: Arabic

Singular		Plural	
qalb	'heart'	*qulūb*	'hearts'

(Eifring and Theil, 2005)

Strategies to word and phrases formation are further complicated due to their vague definitions in linguistics. For example, a word is already a controversial object, both within one language and cross-linguistically. In the modern European writing system, a blank space is often used to indicate boundaries between words. Yet, this conventional spelling can be sometimes misleading and ambiguous to identify words. Sometimes boundary symbols other than a blank space are used. For example, in French, a hyphen is used with object pronouns for form imperative, e.g. *donne-le-moi* 'give it to me'. Sometimes the same element is spelled differently under different circumstances. For example, in German, the infinitive marker is spelled separately in most cases (e.g. *zu bringen* 'to bring'), but together with the verb when it is preceded by a prefix such as *ein-* 'in' (e.g. *einzubringen* 'to bring in'). The rules for orthographic word division are, to some extent, simply a traditional convention in languages with a long written history. Not all writing systems indicate word boundaries. In Chinese, for instance, there are never blank spaces between characters.

When developing language technologies, e.g. machine translation, question answering, speech recognition, the objective is to enable a computer to process and generate text as precise as humans do. For decades, NLP has been focusing on the development of such systems for English. In the age of digitalization, more and more data becomes available for other languages of the world. This led to a growing interest to expand NLP technologies to new languages. However, the systems developed for English are often not easily portable to other languages. One reason for this is the lack of task-specific data, despite growing amounts of written text data on the web. For example, machine translation requires aligned sentences in two languages. The other reason is a drastic variation of language systems, illustrated above. Systems developed for English would not be easily adaptable to other languages which use, for example, different writing scripts or complex word formation processes. One way to address such challenges is the integration of **upstream processing** methods into language technologies.

Many of upstream methods involve the processing of text with the purpose of extracting linguistic information. For example, this could be achieved by segmenting word forms into morphemes, annotating word forms with part-of-speech tags or assigning a syntactic tree to a phrase. Such annotation is then integrated into a downstream system. Upstream text processing approaches are developed as separate NLP tasks, while their further integration into systems, upstream or downstream, is

yet a different avenue. In this thesis, we tackle both directions by considering the task of **subword segmentation**. We develop our models with the goals of a) improving their **performance** compared to previous solutions; b) making models more **interpretable** by aligning its decision process more closely to human intuition; c) proposing solutions which are **portable** between tasks and languages. We address these challenges by focusing on **multi-level modelling**, which allows the extraction and integration of information found at different text organization levels.

1.2 Subword Segmentation as Upstream Text Processing

One of the essential questions in designing NLP systems is what to take as an atomic unit: separate symbols, orthographic words or subwords. Subwords are understood as clusters of characters and can range from individual characters to word segments and include individual words. Given the variety of language scripts and cross-linguistic differences to morpho-syntactic strategies, discussed above, this aspect of the system design is not always portable from one language to another. The choice of the atomic units should balance between two objectives. On the one hand, the system should capture well regularities of languages and generalize well when processing unseen words. On the other hand, it should be efficient in terms of parameter estimation and employment at test time.

An optimal processing unit level has been a vivid topic in both statistical and neural models in NLP. For example, in the machine translation (MT) domain, a statistical framework known as statistical machine translation (SMT) generally modeled orthographic words as atomic units. This inevitably results in the loss of information that could be extracted from character sequences within words. For example, frequent character sequences that mark the plural form of nouns, illustrated in Ex. (1), would not be identified in unseen words. This could lead to a wrong prediction on a sentence level, when there is an agreement between a noun and other words in a sentence in plural marking, i.e. other words get also marked for plural to agree with a noun. Linguistic features such as lemmas, morphological and syntactic information have been used in an attempt to overcome such problems and to improve the performance of SMT (Koehn and Hoang, 2007), resulting in factored machine translation models. Character-level SMT models have been shown to work well when translating between closely related languages, e.g. Spanish and Catalan (Vilar, Peter, and Ney, 2007), Norwegian and Swedish (Tiedemann, 2009).

The search for optimal units continues with the shift of NLP paradigm, from statistical to neural one. This change in paradigm started with the introduction of feed-forward neural networks to the fundamental NLP task of language modelling (Bengio et al., 2003). After these first steps in neural language models, neural networks usage in NLP stagnated for a decade, until more powerful algorithms and computing power finally enabled efficient parameter estimation in such models. Subsequent work of

Mikolov et al. (2011) on applying recurrent neural networks (RNN) to language modelling popularized the usage of RNN language models (RNNLM) in NLP. RNNLM laid the foundation for the RNN encoder-decoder (ED) system (Cho et al., 2014a; Sutskever, Vinyals, and Le, 2014; Bahdanau, Cho, and Bengio, 2015) which replaced statistical approaches to MT in NLP. Initially, ED systems used words as processing units. Similar to SMT, to overcome the problem of processing unseen words, there have been research efforts to explore hybrid character-word ED architectures (Ling et al., 2015b) and fully character-level ED systems (Lee, Cho, and Hofmann, 2017). Such frameworks provide flexibility to process unseen words. However, they result in longer sequences which leads to computational challenges. Introduction of **subword segmentation** method with byte-pair encoding (Sennrich, Haddow, and Birch, 2016), has marked a 'subword revolution' in NLP: such approach provides a balance between handling out-of-vocabulary (OOV) and computational efficiency.

The community has settled on using subwords as atomic units in ED systems. This practice was adopted in the later extensions of ED systems, such as transformer model (Vaswani et al., 2017) and contextual language models (for example, BERT of Devlin et al. (2019)), which are ubiquitous in NLP at present. However, there is conflicting evidence on which approach to the subword segmentation itself is better: whether subword segmentation should be learned using linguistically inspired approaches or purely data-driven ones. In this thesis, we consider both approaches and apply them in different settings. First, we propose a method to improve linguistically motivated subword segmentation. Then, we test our method's portability to another upstream task of writing normalization where segments are more implicit. Finally, we propose integrating data-driven subword segmentation to improve performance of yet a different upstream task of inflection generation.

1.2.1 Linguistically Motivated Subword Segmentation

In the first study of this thesis, we develop a linguistically motivated approach to subword segmentation by tackling the task of canonical morphological segmentation, illustrated in the next example:

(2) Canonical Morphological Segmentation (English)

Input Word		Segmentation
jumped	▷	*jump*|*ed*
tied	▷	*tie*|***e****d*
*sa**i**d*	▷	*sa**y***|***e****d*
exceptionably	▷	*except*|*ion*|*ab**le***|*ly*

The goal of canonical segmentation is to break a word up into its constituent morphemes. The crucial difference to another popular task of surface segmentation is that resulting segmentation contains morphemes in their canonical form. The canonical form is selected in the annotation data to represent all possible morpheme variations

due to morphophonological changes on morpheme boundaries. Such changes often result in deleting or inserting characters in the surface realization of a morpheme, and canonical segmentation restores such changes. We highlight morphophonological changes in bold in Ex. 2. For instance, the canonical form of morpheme *-ed* marking past tense in English corresponds to its surface realization *-ed* as in the word *jump-ed*, or *-d* as in the word *sai-d*. Prepossessing text with canonical segmentation allows a downstream system to recognize that substrings *-d* and *-ed* share the same linguistic function.

Surface segmentation is often solved with unsupervised methods and, therefore, benefits from using abandon raw text data to estimate parameters for such methods. Contrary to this, the canonical segmentation task only disposes of manually annotated corpora. In this thesis, we develop methods that can efficiently learn segmentation from a small annotated corpus. Such data sources have become increasingly available in the linguistic community across more and more languages.

1.2.2 Subword Segmentation as a Part of Text Processing

We adapt subword segmentation to improve other upstream text processing methods in two separate studies. In one study, we tackle the task of writing normalization. In this task, words in a non-canonical language, typical of speech transcription or computer-mediated communication, are mapped to their standardized writing. We illustrate writing normalization in the next example:

(3) Writing Normalization

Source form (Swiss German)		Normalized form (Standard German)
viil 'much'	▷	*viel*
viu 'much'	▷	*viel*
vill 'much'	▷	*viel*
hämmers 'we have it'	▷	*haben wir es*

where three dialectological variants of a Swiss German word 'much' are mapped to their canonical word from in Standard German. Considering the last line in Ex. 3, we notice that subword segmentation and word normalization exhibit a common feature. Specifically, the word *hämmers* is realized as a sequence of three separated words in Standard German. Such phenomenon corresponds to an implicit segmentation and motivates adaption of the approaches for canonical segmentation where segments on the target side correspond to words. We exploit this fact and test portability of our methods developed for canonical segmentation to a task of writing normalization.

In another study, we propose an integration of word segmentation to improve on morphological inflection generation. The upstream task of inflection generation is illustrated in the next example:

(4) Inflection Generation Task (Italian)

Lemma		Inflection Tags		Inflected Form
chiudere	+	V;IND;PRS;3;PL	▷	*chiudono* '(they) close'
giocare	+	V;IND;PRS;3;PL	▷	*giocano* '(they) play'

The task's goal is to map a lemma to its target inflected form, given a sequence of target inflection tags. We argue that lemma's segments provide a linguistically motivated signal useful for solving such task. For example, the choice between inflection suffixes *-ono* and *-ano* in Italian (Ex. 4) depends on whether the lemma ends on *-ere* or *are*. This motivates us to explore the integration of subword segmentation into inflection generation model. To this end, we adapt methods for surface subword segmentation.

1.3 Multi-level Modelling

The upstream tasks, we consider in this thesis, have been recently successfully solved with encoder-decoder methods run on character-level. Choosing characters as atomic units in solving upstream tasks results in computationally efficient solutions due to two reasons. First, the amount of parallel data available for these tasks is usually much smaller than in the downstream tasks which can be solved with ED systems, e.g. machine translation. Therefore, running the ED system in such limited data setting on characters instead of subwords allows exploiting regularities in sequence transformations more fully. Second, upstream tasks are usually defined on a word or short phrase level. Hence, a model has to learn to transform sequences of characters of relatively shorter length compared to downstream sequence transformation tasks. Character-level modelling is very flexible in capturing regularities of local character transformation. However, we argue that regularities can be found in higher-level units too and can provide a useful signal for solving upstream tasks.

1.3.1 Integrating Signal from Two Structural Levels: Subwords vs Context

In this thesis, we propose novel solutions to solve three different upstream tasks. All our solutions are multi-level: we take a character-level ED model as a starting point and propose hierarchical modifications to such system for each task. Our modifications integrate a signal extracted from the structural levels of text organization higher than characters.

As the first type of higher-level information, we consider regularities found in **subword** sequences. Specifically, a data example in each task involves the transformation of a source character sequence corresponding to one orthographic word into a target character sequence corresponding to one or more orthographic words. We

consider different types of subword sequences which can be extracted from such character sequences for each upstream task. We illustrate them in Ex. 5 where we use previously introduced examples of all three tasks:

(5) Subword-level signal

a. Canonical Morphological Segmentation (English)

Input word		Segmentation
exceptionably	▷	*except* | *ion* | *able* | *ly*

b. Writing Normalization

Source form (Swiss German)		Normalized form (Standard German)
hämmers 'we have it'	▷	*haben* *wir* *es*

c. Inflection Generation Task (Russian)

Lemma		Inflection Tags		Inflected Form
chiud | *ere*	+	V;IND;PRS;3;PL	▷	*chiudono* '(they) close'

The highlighted units in this example correspond to a) morpheme sequences of word segmentation (5a); b) substring sequences of word segmentation (5c); and c) orthographic word sequences of word normalization (5b). We identify subword signal either on the target side of sequence transformation (5a and 5b) or on the source side (5c).

Besides the subword-level signal, we propose integrating higher-level information corresponding to the use of an input word in a context. Such **context** signal goes beyond word level and allows to address ambiguity in text processing: same word form can be processed differently depending on a concrete sense in which this word is used. We explore the integration of context information for the task of writing normalization. For extraction of the higher-level signal of context we consider a) part-of-speech (PoS) tags and b) words surrounding an input word when it is used in a context represented by an utterance.

For each task, we design experimental studies in order to test whether such multi-level methods improve performance over pure character-level ED systems.

1.3.2 From Performance to Model Interpretability, and back

Upstream processing tasks are helpful not only in improving language technologies. Developing high-performing methods on these tasks can aid and scale-up fundamental linguistic research. Linguistic analysis often relies on a small manually annotated corpus created by field linguists. These resources can be used to develop upstream processing models. Such methods allow automatic annotation of text and therefore, provide more material for linguistic research.

The exchange between the two fields of research is beneficial on a broader scale. Beyond providing annotated data, insights from theoretical linguistics can be further

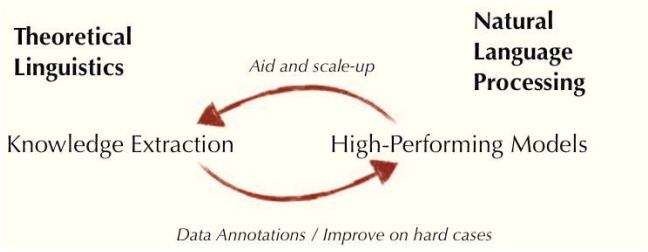

FIGURE 1.1: Cyclical exchange between NLP and Theoretical Linguistics.

fed into new NLP methods. The resulting cyclical interplay between NLP systems and theoretical linguistics is schematically illustrated in Fig. 1.1. The exchange becomes even more desirable in light of the recent shift to neural models in NLP. On the one hand, neural models have boosted the performance of language technologies. On the other hand, they became much more obscure than statistical modelling where the integration of linguistic features provided control over the model behaviour, to a certain extent. However, the nature of neural networks, i.e. their similarities to the human brain structure, together with high performance, have recently stimulated research efforts in interpreting what linguistic knowledge emerges in neural network models. Understanding how neural models process and generate text can provide new opportunities and material to test linguistic theories. And at this point, insights from theoretical linguistics can be integrated into the model. In particular, aligning model's decision process on a particular NLP task to human intuition could be a possible solution to improve on hard cases.

In this thesis, we demonstrate that such cyclical nature emerges in neural upstream processing too. Our motivation to introduce multi-level modelling to improve character-level solutions on three upstream tasks comes from a well known linguistics result. According to the double articulation principle (Martinet, 1967), single phones are uninterpretable to human, while clusters of them form a mental linguistic representation of meaning in the speaker's mind. In writing systems, these clusters transform into morphemes that are traditionally considered the smallest information units. Therefore, in order to align a neural learning process to human intuition, we would like the neural model to base its decision by analyzing clusters of characters. To achieve this goal, we develop approaches for integrating a higher-level signal into a character-level neural model. We come back to theoretical linguistics by developing interpretability methods to analyze what such multi-level model learns. Our interpretability methodology for extracting knowledge of inflection morphology from neural networks provides new opportunities for testing theories in theoretical morphology scaled to languages of diverse typology. In particular, it allows employment of resources which are traditionally not used for theoretical studies of inflection: crowd-sourced wiktionary data and high-performing neural inflection models.

1.4 Outline

This thesis proposes multi-level approaches for integrating a higher-level signal into character-level ED models. We show that our hierarchical methods a) improve performance on three upstream processing tasks; b) make a character-level ED model more interpretable. We describe and test our multi-level models for morphological segmentation in Chapter 4, for writing normalization in Chapter 5 and for inflection generation in Chapter 6. We perform model interpretation study within one language in Chapter 5 while we propose a method to interpret what model learns scalable to many languages in Chapter 6. We focus on developing solutions which are portable between tasks or languages. In particular, we test the portability of our methods developed for the morphological segmentation in Chapter 4 to the task of writing normalization in Chapter 5. To assess how our methods generalize to the languages of different typology, we analyze our methods on several languages in Chapter 4 and Chapter 6. This thesis is organized as follows:

Chapter 2 In this chapter, we introduce upstream text processing and its objectives. We start by presenting some of the key concepts of morphology and how languages vary with respect to their morphological processes. This prepares the terminology required to introduce three upstream processing tasks considered in this thesis. We will also rely on this terminology when describing linguistic intuition behind our methods in the later chapters. For each task, we contrast its role in computation linguistics to that in general linguistic research.

Chapter 3 This chapter introduces background information on methodology. We start with the description of statistical and neural approaches to language modelling. We then explain how the second approach gives rise to the encoder-decoder neural paradigm used for sequence transduction. We build on both approaches, language modelling and encoder-decoder framework when developing our multi-level methods in later chapters.

Chapter 4 In this chapter, we propose a multi-level encoder-decoder model for learning canonical morphological segmentation. The method consists of two components: character-level encoder-decoder model and a language model over canonical segments. We combine the two components during the decoding stage through our novel synchronized decoding approach. Our method is designed to take advantage of a small annotated corpus. In particular, the goal of its design is to learn from the same parallel data by combining a signal coming from two levels of text organization, characters and segments.

We test our model in two settings. First, we test the impact of integrating additional higher-level language model for three different languages. We develop an evaluation procedure to assess the proposed segmentation algorithm's generalisation

across different languages by introducing specific categories of unseen words. Although our method is designed to take full advantage of small parallel corpora, it is possible to integrate extra language data. Such data can be applied to provide more training material for the higher-level language model. In the second experiment, we test our models in the setting where we integrate additional structured language data in the form of dictionaries. Some of the experiments in this study were published in Ruzsics and Samardžić (2017).

Chapter 5 In the next chapter, we argue that our synchronized decoding method proposed for canonical segmentation can be portable to another upstream task where segmentation is more implicit. We demonstrate such portability by applying the synchronized decoding method in our next study. In this study, we tackle the upstream task of writing normalization. We propose to solve this task with a multi-level approach where we integrate two modifications into a plain character-level encoder-decoder system. The first modification is an adaptation of our synchronized decoding, which allows integration of an additional word-level language model on the decoder side. The second modification incorporates context signal that can be extracted from an utterance in which a word is used. We propose several solutions for context integration: a) in the form of part-of-speech (PoS) tags; b) as a hierarchical neural language model on the encoder side trained over sequences of words in context; c) as a combination of the two.

In our first experiment, we design quantitative and qualitative analyses to explain our model performance in two ways. First, we test a complementarity hypothesis of our two modifications where we take into account only one context model which uses the PoS tag signal. Second, our analysis is designed to assess whether a context signal in the form of PoS tags is sufficient to resolve cases of ambiguity in the task. This work has been published in Ruzsics et al. (2019).

We design our second experiment to test two hypotheses concerning the context integration: a) whether integrating context signal as a hierarchical neural language model on the encoder side is more efficient in targeting ambiguity problem than the PoS tag signal; b) whether a combination of two types of context signal provides further improvements in solving ambiguity. Regarding the synchronized decoding, we perform an additional portability test in our second experiment where we add a setting with extra monolingual target-side data.

Chapter 6 In the next study, we develop a methodology for extracting knowledge of inflection morphology learned by neural models. To this end, we propose a model trained on the task of inflection generation and a pattern extraction method to analyze this model. We provide linguistically motivated argumentation that in order to align our interpretability method more closely to human intuition, we require integration of subwords to our methodology. To achieve this objective, we propose a multi-level modification to a standard encoder-decoder model. In addition to representing

the input word as a sequence of characters, we consider a representation where it is segmented into subwords. The higher-level signal from a sequence of subwords is integrated into the neural model through an additional subword-level attention mechanism.

The interpretability method relies on both attention mechanisms of our model, standard character-level attention and our proposed subword-level attention, in order to extract inflection patterns from learned attention components. The goal of such approach is to represent the knowledge learned by a neural model as a 'database' which can be queried for an inflection category of interest and return two types of patterns: a) linguistic rules and b) data examples corresponding to a rule.

In our experiments, we test the performance impact of our proposed subword-level attention mechanism. Coming back to the cyclical nature of NLP, discussed earlier, we demonstrate how our methodology can be used for querying inflection patterns. We conduct our experiments on three typologically different languages. This allows us to assess how portable our approach to encompass a variety of cross-linguistic morphological processes for inflections. This work has been published in Ruzsics et al. (2021).

Chapter 7 In this chapter, we first summarize our main contributions and findings. After that, we give pointers for future research directions.

Chapter 2

Upstream Text Processing

Languages of the world use different strategies to convey information. First of all, they choose different basic units of sound and writing systems. Next, there are different ways to combine them into higher-order units. Such combinations are at the interplay interplay between phonetic, morphological and syntactic systems of a language. Each system governs which combinations of units are acceptable in a language. For example, phonetics governs how sounds combine into the smallest meaningful units which correspond to morphemes in writing. Morphology describes how morphemes form words.[1] Finally, syntax describes the rules governing the combination of words into utterances. These three layers are not entirely separated: the interactions are modelled by morphophonological and morphosyntactic rules. All those components which describe the language system as a whole are usually found in language grammars.

While language systems of the world are marked by considerable diversity, there are empirically attested commonalities. One of them is the Zipf's Law (Zipf, 1935): relatively few word forms are used frequently while most word forms occur rarely. Larger samples of text reinforce such patterns exhibited already in smaller samples, by introducing progressively fewer new forms. This systematic form of data sparsity results in a challenge for downstream NLP systems: how to learn grammatical structure efficiently and how to generalize models to unseen word forms? This is where upstream processing methods come into play. In this thesis, we develop models for three such methods, each of which forms a separate NLP task described in this chapter.

One step towards tackling the problems caused by the Zipf law is the search for optimal processing units: instead of words, should NLP systems model text as sequences of characters or subwords, i.e. string of characters smaller than words? For example, in machine translation, Sennrich (2017) showed that while characters are more effective than subwords for production of novel words, they perform worse in capturing long-distance dependencies, e.g. morphosyntactic agreement. This result was recently reinforced in the work of Belinkov et al. (2020). The ultimate choice of units

[1] The notion of a word (as well as many other linguistic units) is a fuzzy concept due to large cross-linguistic variation of language systems. In this thesis, we will refer to words, or word forms, in the sense of orthographic words: strings of characters separated by spaces. Such definition is most commonly adopted in NLP.

depends on a) the overall system performance and b) efficiency, i.e. how computationally expensive it is to estimate the parameters of models. NLP community currently regards subwords as the optimal processing units. Upstream processing for subword segmentation became a norm. However, there is still no consensus on whether the subword segmentation approaches should produce linguistically motivated segments or not. In this thesis, we work with a variant of subword segmentation task - canonical segmentation - where segments are linguistically motivated.

Data sparsity becomes even a more significant problem when there is no standardized writing. As a result, there is a high degree of variation in writing the same word. This factor makes it challenging to develop NLP tools for processing spoken dialects or historical texts. In order to reduce data sparsity in such cases, text normalization is usually applied. Writing normalization maps orthographic words to their canonical form, i.e. their form in a language with a standardized orthography. In our work, we consider the task of writing normalization for Swiss German dialects.

As NLP keeps expanding its frontiers to encompass more and more languages, data sparsity becomes a more pressing issue. In the case of low-resourced languages, the search for optimal processing units for downstream applications has to be combined with other forms of upstream processing. In this case, extra upstream processing aims to add more explicit modelling of language's linguistic structure. For example, downstream applications can benefit from injecting a knowledge of inflection morphology. One way to achieve this is through the integration of methods for inflection generation.

In this thesis, we focus on the three upstream methods: canonical morphological segmentation (Section 2.2.1), writing normalization (Section 2.2.2) and morphological inflection generation (Section 2.2.3). We develop models for all three tasks in a supervised setting. Therefore, we formulate them as supervised tasks after introducing supervised machine learning terminology in Section 2.2. Subword segmentation can also be formulated in an unsupervised setting. We resort to unsupervised subword segmentation methods when developing inflection generation models and provide more details on such methods later in Chapter 6. The two of the considered tasks - canonical segmentation and inflection generation - are directly related to modelling morphology of a language. The notions used in morphology can vary from one linguistics textbook to another. Therefore, we introduce the terminology in Section 2.1, which we follow in this thesis in order to formally describe the tasks as well as build methods for them in the later chapters.

Reflecting the dual nature of computational linguistics, all three upstream processing methods introduced in this chapter, are helpful not only for practical applications but also for fundamental linguistic research. On the one hand, upstream NLP allows automatic creation of linguistic annotation, given a limited amount of resources annotated by humans which are costly to produce. On the other hand, data and models for such processing tasks are useful for testing linguistics theories. For example, the same Zipfian distribution that makes building NLP downstream systems difficult is

also at play in morphology: in the same way as these systems have to process and generate unseen word forms, how does a speaker produce a different inflected form for a previously unseen word? How to measure the difficulty of such learning process, and what is its impact on measuring the complexity of a morphological system cross-linguistically? The examples and references for using the three considered NLP tasks in practical applications and linguistic research are discussed in their respective sections.

2.1 Background on Morphology

Morphology studies internal structure of words. As opposed to phonology, which studies internal structure in relation to sequence of sounds, morphology deals with systematic covariation in the form and meaning of words. Concretely, morphology seeks to identify morphological patterns which comprise a) finding the smallest meaningful constituents of words; and b) the rules how these constituents can be combined to form words. Next, we introduce basic terminology in morphology[2], starting with notions of words and morphological relations between words described by morphemes. We then move to the description of types of morphemes and more general morphological processes.[3]

Word-forms vs Lexemes When a dictionary is made, not every word is given its own entry. For instance, the words *read*, *reading* and *reads* are different words, a dictionary would contain only a single entry *read*. A word in an abstract sense is called a **lexeme**. For example, lexeme `read` represents the core meaning shared by forms *read*, *reading* and *reads*. Lexemes are abstract entities that have no phonological form of their own. By contrast, a **word-form** is a word in a concrete sense. It is a sequence of sounds that expresses the combination of a lexeme and a set of grammatical meanings appropriate to that lexeme. For example, word form *reads* combines the lexeme `read` with a grammatical function of third person singular present tense. A particular word form that is chosen by convention to represent a lexeme is referred as **lemma**: for example, lexeme `read` is represented by a word form *read*. When a word form is used in some text or in speech, that occurrence of the word form is often referred to as a **word token**.

Inflection vs Derivation Morphological relations between different words are usually divided into two broad categories: inflection and word formation. Inflection morphology deals with the relationship between word-forms of a lexeme. The set of word-forms that belongs to a lexeme is often called a **paradigm**. Paradigms can

[2]We follow closely Haspelmath (2010)

[3]In this section, the examples for Russian, German and Finnish are constructed by the author and her colleagues, examples for other languages which taken directly from Haspelmath (2010) are not explicitly referenced, the rest of the examples are taken from grammars with an explicit reference to it.

be represented by a grid where each cell is filled with an inflected form for a given lexeme. The paradigm of Russian noun lexeme *ruka* 'hand' is illustrated in Ex. 6. Traditionally, the paradigm of a noun or adjective is called its declension and that of a verb its conjugation.

(6) Complete Paradigm of Russian *ruka* 'hand'

Case | *Number*	Singular	Plural
Nominative	*ruk-a*	*ruk-i*
Genitive	*ruk-i*	*ruk-ø*
Dative	*ruk-e*	*ruk-am*
Accusative	*ruk-u*	*ruk-i*
Instrumental	*ruk-oj*	*ruk-ami*
Locative	*ruk-e*	*ruk-ah*

Different lexemes may also be related to each other, and a set of related lexemes is sometimes called a word family. For example, lexemes `read`, `readable` and `reader` belong to one word family. The relationship between lexemes of a word family is a focus of derivational morphology.[4]

Some words belong to two (or more) word families simultaneously. For instance, the lexeme `firewood` belongs both in the family of `fire` and in the family of `wood`. Such relationships are called compounding, and lexemes like `firewood` are called compounds. Compounding is often grouped together with derivation under the category of word formation (i.e. lexeme formation).

Morphemes: Abstract vs Concrete Forms Morphemes can be defined as the smallest meaningful constituents of a linguistic expression. In both inflection and derivation, morphemes have various kinds of meanings and forms. In the simplest cases, words can be easily segmented into morphemes, i.e. broken up into individually meaningful parts: *re + read*, *read + able*. However, both the shape and the meaning of a morpheme can be much more abstract. To cover the variety of **forms**, it is common to refer to morphemes as **morphological patterns**. In abstract terms, a morphological pattern defines a **morphological operation** applied to a **base** (or, stem, especially in inflection). We will look at the examples of morphological patterns starting with **affixation**.

An affix is a prototypical morpheme which is a segmentable part of a word. It is a bound morpheme in a sense that it cannot occur as a word-form by itself. The process of affixation comprises two restrictions: selection and position. Selectional restriction defines the type of words it can attached to, e.g. in English affix *-able* attaches only to verbs. The base in affixation process is the part of the word that an affix is attached, e.g. in English *read-able* the base is *read*. Positional restriction defines how it attaches

[4]The exact distinction between inflection and derivation can be fuzzy. For instance, does the word-form *nicely* belong to the lexeme `nice`, or does it represent a lexeme of its own (`nicely`), which is in the same word family as *nice*?

to the base: affixes that follow the base are called **suffixes** (e.g. the English *-able* or the Russian case and number suffixes in Ex. 6), and affixes that precede it are called **prefixes**, (e.g. *re-* in English *re-read*). Bases can be complex themselves. For instance, in *readability*, *-ity* is a suffix that combines with the base *readable*, which itself consists of the suffix *-able* and the base *read*. A base that cannot be analyzed any further into constituent morphemes is called a **root**.

Suffixation and prefixation belong to the very common type of morphological patterns called concatenative. **Concatenative patterns** denote linear combination of units and also manifest themselves in compounds. When we talk about **morphemes** in this thesis, we will refer to segmentable parts of the words, i.e. suffixes, prefixes and segmentable roots. All the other morphological patterns are referred to as **non-concatentative**. Already affixation involves non-concatenative patterns. For example, **infixes** occur inside the base, e.g. Tagalog *s-um-ulat* 'write' where infix *-um-* is attached to base *sulat*. A **circumfix** occurs on both sides of the base, e.g. German *ge-fahr-en* 'driven' where circumfixg *ge-...-en* is attached to base *fahr*.

While in the cases of affixation it is possible to segment a morpheme inside of a word, there exist much more abstract non-concatenative patterns. One important class of non-concatenative patterns is **base modification** (or stem modification/alternation). This is a collective term for morphological patterns in which the shape of the base is changed without adding segmentable material. A common type of base modification pattern results from changing the place of articulation. For example, in German one way of forming the plural of a noun is by fronting of the stem vowel, i.e. changing the place of articulation so that the vowel is pronounced more towards the front of the mouth. As illustrated in Ex. 7, a back vowel of the singular form (e.g. [ʊ], [a:], [ɔ], spelled *u, a, o*) is replaced by a front vowel (e.g. [ʏ], [e:], [ø], spelled *ü, ä, ö*).

(7) Base Modification: German

Singular	Plural	
Bruder	*Brüder*	'brother(s)'
Garten	*Gärten*	'garden(s)'
Vogel	*Vögel*	'bird(s)'

Another common morphological operation is **reduplication**, whereby a part of the base or the complete base is copied and attached to the base (either preceding or following it). It is somewhat a middle case between concatenative and non-concatenative patterns, since it uses a material from the base and involves linear attachment to the base. For example, in Tagalog verbs are inflected for contemplated aspect by preffixing a template CV- to the base, where C is a slot for the first consonant of the base and V - for the first vowel, as illustrated in Ex. 8. Contemplated aspect in Tagalog is used to report an event which has not yet begun.

(8) Reduplication of a CV sequence before the base: Tagalog

Singular		Plural	
tapusin	'finish'	*ta-tapusin*	'will finish'
bigyan	'give to'	*bi-bigyan*	'will give to'

(Schachter and Otanes, 1983)

Morphemes: Abstract vs Concrete Meanings In the same way, as the form of a morpheme can be different, its **meaning** can vary from concrete to abstract. Usually meanings of roots and derivational morphemes can be more concrete, e.g. the meanings of the morphemes *wood*, *fire*, *un-*. Other derivational meanings can be more abstract. For instance, the morpheme *-al* in *logic-al* can perhaps be said to mean 'relating to' (cf. *logic-al, mathematic-al, physic-al*), *-able* in *read-able* can be said to mean 'capable of undergoing a process'. One frequent characteristic of derivational patterns is that they commonly change the word-class of the base lexeme – i.e. nouns can be derived from verbs, adjectives from nouns, and so on.

Meanings of inflection morphemes are usually more abstract and describe a grammatical function, or **values of inflection features**. For instance, English verbs express the inflectional values 'present' (e.g. (he/she) walks) and 'past' (e.g. (he/she) walked). Different languages vary quite dramatically in the amount of inflectional complexity that their words exhibit. However, despite all this diversity, the types of inflectional values that are found across languages are surprisingly uniform. For example, nouns and adjectives can express inflection features of number, case, gender and person, while verbs can express inflection features of number, person, tense, aspect and mood. Each inflection feature can have a concrete value: for example, person can be first, second and third; tense can be present, past and future.

Allomorphs: One Meaning but Different Forms One of the most common complications in morphology is that morphemes may have different phonetic shapes depending on the context. When a single morpheme has more than one shape, those shapes are referred to as **allomorphs**. The examples in (9) illustrate affix allomorphy in Finnish and stem allomorphy Russian.

(9) a. Finnish (vowel harmony): all suffixes containing one of the harmony vowels */u, a, o, ü, ä, ö/* have two variants, one for back vowels */u, a, o/* and one for front vowels */ü, ä, ö/*. The back form occurs if there is a back vowel to the left, else the front ending variant occurs.

Singular		Plural	
laulaa	'to sing'	*laula-vat*	'(they) sing'
tietää	'to know'	*tietä-vät*	'(they) know'
lukea	'to read'	*luke-vat*	'(they) read'

b. Russian: when the stem is followed by a vowel-initial suffix, the vowel *o*/*e* is often dropped if it is the last vowel in the stem

Singular		Plural	
lokot'	'elbow-N'	*lokt-evoj*	'elbow-ADJ'
meshok	'bag'	*meshk-i*	'bags'

Allomorphs illustrated in Ex. 9 show only fairly small differences in the shapes of morphemes and are phonologically similar. Allomorphs with this property are called **phonological allomorphs**. Alternations between the form of phonological allomorphs are often described by morphophonological rules. It is often convenient to think about phonological allomorphy in terms of a single underlying representation that is manipulated by rules under certain conditions. The end result, i.e. what is actually pronounced, is the surface representation. For instance, in Russian example 9a, an underlying representation of stem is [lokot'], which after applying the rule described in the example, is alternated to a surface representation [lokt].

Besides phonological allomorphs, morphemes may also have allomorphs that are not at all similar in pronunciation. These are called **suppletive allomorphs**. For instance, the English verb *go* has the suppletive stem *wen* in the past tense (*wen-t*), and the English adjective *good* has the suppletive stem *bett* in the comparative degree (*better*).

In the next section, we will use the described terminology in order to introduce upstream text processing methods considered in this thesis.

2.2 Upstream Processing as Supervised Machine Learning

In a supervised machine learning setting, the goal is to find a function that 1) maps an input to an output based on a set of examples; 2) generalizes well, i.e. maps correctly input to outputs for new (unseen) examples. The set of example which is used to infer the function is referred to as training data, while the process of finding this function and setting its parameters as training.

In the next subsections, we will formulate three upstream processing methods as supervised NLP tasks. For each task, we will provide concrete examples of input and output data pairs. The technical aspects of machine learning, i.e. how to find a function which generalizes well, will be explained in more details in the next chapter.

2.2.1 Canonical Segmentation

In theoretical linguistics, in addition to descriptive language grammars, analysis of internal word structure is commonly performed with Interlinear Glossed Text (IGT). IGT is a rich data type where text is annotated with semantic and grammatical properties of a language. IGT consist of a source language phrase, a translation of that phrase into the language of the target audience, such as English, and glosses

for each source morpheme. The glosses highlight the morphological and syntactic features of the source language. Ex. 10 shows an IGT from a Chintang [ctn] (Kiranti, Nepal) corpus:

(10) cuwa thaptakha!

cuwa	*thapt*	*-a*	*-khag*	*-a*
water	move.across	-IMP	-see	-IMP[2sS]

'Bring some water over here!'

(Stoll, Mazara, and Bickel, 2017)

In this example, the first line is the source phrase, the second one contains the morpheme segmentation of the words in the phrase, the third line shows glosses assigned to each morpheme, and the last one is the translation line. The glosses can refer to grammatical information, i.e. imperative mood (IMP) or agreement of imperative mood with second person singular subject (IMP[2sS]), or to lexical meaning, i.e. 'water'. The glossing convention has been standardized by Leipzig Glossing Rules.[5]

With the spread of natural language processing to a wider range of languages, building resources and learning internal word structure becomes increasingly important for developing practical applications. A particular type of such analysis, termed morphological segmentation in NLP, aims at automatically discovering meaningful parts - morphemes - in words. Integrating this knowledge into systems has been shown helpful in reducing data sparsity for morphologically rich languages and overcoming limited data amount for under-resourced languages. The upstream task of the morphological segmentation can be defined in two ways, illustrated in the following example with the Chintang verb from Ex. 11:

(11) a. Surface segmentation:

Input Word		Segmentation
thapta**kh**a	▷	thapt-a-**kh**-a

b. Canonical segmentation:

Input Word		Segmentation
thapta**kh**a	▷	thapt-a-**khag**-a

The term *surface* segmentation refers to the analysis where the input word is segmented into substrings without any further string transformation. In this way, the words are segmented into allomorphs. This definition is most widely applied in computational processing; it is, however, too simplistic for the majority of languages. It does not allow, for instance, to identify *-es* in *bus-es* and *-s* in *car-s* as two variants of the same English plural marker.

[5] https://www.eva.mpg.de/lingua/resources/glossing-rules.php

More recently, the term *canonical segmentation* was used by Cotterell, Vieira, and Schütze (2016) to refer to the same definition that was termed *morpheme* segmentation by Creutz and Linden (2004). In this case, a more abstract internal word structure is learned by transforming the resulting substrings for each identified allomorph into their canonical forms. Canonical form is selected in the annotation data to represent all possible variations of a morpheme. As discussed in Section 2.1, such changes are often phonologically conditioned and are described by morphophonological rules. The resulting changes can be represented as deletion or insertion of characters in the surface realization of a morpheme and canonical segmentation restores such changes. We illustrate canonical segmentation for words in English and highlight such morphophonological changes in bold in Ex. 12.

(12) Canonical Morphological Segmentation (English)

Input Word		Segmentation
exceptionably	▷	*except|ion|ab**le**|ly*
communicable	▷	*communica**te**|able*
jumped	▷	*jump|ed*
tied	▷	*tie|**e**d*
*step**p**ed*	▷	*step|ed*
*sa**i**d*	▷	*sa**y**|**e**d*

For instance, canonical form of morpheme *-ed* marking past tense in English corresponds to its surface realization *-ed* as in the word *jump-ed* or *-d* as in the word *sai-d*. A derivational suffix *able* forming adjectives in English can be realized on surface as *ab*, e.g in the word *exceptionably*. Yet, allomorphy can happen in roots too. For example, deleted substring *te* is restored in canonical segment *communica**te***.

In terms of variety of morphological patterns used by languages (Section 2.1), morphological segmentation task is appropriate for processing words with concatenative patterns. While this may sound restrictive to encompass great variety of encoding strategies used by languages, concatenative patterns seem to be the most frequent cross-linguistically: "Concatenative patterns are more common than non-concatenative patterns in part because the language structures that are the historical sources of concatenative patterns are more common than the language structures that are the historical sources of many non-concatenative ones." (Haspelmath, 2010). Indeed, from the practical point of view, both types of morphological segmentation proofed to be useful in downstream applications across many languages in terms of performance impact and in particular, in addressing data sparsity problem. In statistical machine translation, Dyer, Muresan, and Resnik (2008) integrate segmentation of Arabic and Chinese into decoding process to improve translation into English. In speech recognition, Creutz et al. (2007) use morpheme-based language model to improve performance in four morphologically rich languages: Finnish, Estonian, Turkish and Arabic. More recently, using word segments, instead of words, as processing units

has been a major breakthrough (Sennrich, Haddow, and Birch, 2016) at addressing the rare words problem in the end-to-end neural encoder-decoder systems.

The task of morphological segmentation is traditionally approached using finite-state technology, such as OpenFst library (Allauzen and Riley, 2012) and OpenGrm Thrax Grammar Compiler library (Roark et al., 2012). Building a finite-state systems heavily relies on hand-crafted rules. A number of data-driven algorithms have been proposed for surface segmentation in unsupervised setting: they work without supervision and induce, from nothing more than raw text, plausible morpheme segmentations for the words occurring in the text. In this thesis, we focus on developing methods for the task of canonical segmentation in a supervised setting (Chapter 4), where we take advantage of small annotated corpora of the form in Ex. 10 increasingly available in linguistic community. In Chapter 6, we apply surface segmentation algorithm learned in unsupervised setting to improve on the task of morphological inflection generation. This choice of the segmentation algorithm is justified by its high portability to languages with no available resources yet for learning morphological segmentation.

In the context of theoretical linguistic research, it becomes particularly important to be able to segment a wide variety of languages, for which the available data sets consist of small, annotated corpora. Indeed, the task of canonical segmentation is an important step towards producing IGT annotation automatically: IGT is commonly produced in a pipeline approach where the step from annotating morphemes with glosses can be cast as a tagging task (Samardžić, Schikowski, and Stoll, 2015; McMillan-Major, 2020), while the task of identifying morphemes has to handled separately. Automatically produced IGT corpora will greatly support filed linguists in language documentation and language learning, while in the same time, it will provide a valuable resource for learning internal word structure in computational linguistics. Additionally, there is a growing interest in automatic learning of morphological segmentation for the purpose of theoretical language comparison (Bentz et al., 2016).

2.2.2 Writing Normalization

Largely influenced by the work on English and other Indo-European languages with a strong orthographic tradition, the NLP pipeline typically requires standardized text as input. Recently, however, text processing has extended to non-standard varieties, including historical texts, transcribed spoken language and computer-mediated communication (CMC) such as blogs, comments, social media posts, messaging. Modern NLP is also increasingly multilingual, starting to address dialects and languages that have no writing standard at all.

What is characteristic of non-standard text is a non-uniform way of writing the same word types (e.g. *u* instead of *you* in English). While this might appear as a marginal stylistic variation in English, it is a substantial feature of less standardized

varieties. In order to facilitate adaption and creation of NLP tools to process non-standard language varieties, there is an increasing need for writing normalization. Such text processing maps different variant of the same word type to a single string.

In this thesis, we will develop and test our methods for text normalization using Swiss German data. Swiss German refers to a range of German varieties spoken in the Northeastern two thirds of Switzerland. Swiss German dialects are widely used in speech, while standard German is used almost exclusively in written contexts. Despite the preference for spoken dialect use, written Swiss German use has become popular in electronic media. This increased use of spoken dialects in the written domain resulted in accumulation of language material and an interest in automatic processing. Given the non-standard nature of written Swiss German, and the high degree of variation that characterizes it, the need for text normalization, becomes immediately evident.

Swiss German test normalization, illustrated in Ex. 13, addresses the orthographic inconsistency due to several factors. First of all, it is due to the lack of a standardized spelling. This is further complicated by the strong regional and intra-speaker variation. All these factors result in the numerous local variants of the same word. For example, the word *viel* ('much') can appear as *viel*, *viil*, *vill*, *viu*, and many other potential variations.

(13) Writing Normalization

Source form (Swiss German)		Normalized form (Standard German)
viil 'much'	▷	*viel*
viu 'much'	▷	*viel*
vill 'much'	▷	*viel*
hämmers 'we have it'	▷	*haben wir es*

In our experiments, we will consider several resources for Swiss German normalization. ArchiMob corpus (Samardzic, Scherrer, and Glaser, 2016) provides normalization of transcribed videos. These recordings cover all Swiss German dialects of Switzerland. Apart from the dialectological factor, the lack of standardized writing often leads to additional variation in the expert transcription of speech. A set of transcription recommendations, proposed by Dieth (1986), is often used in expert transcriptions. However, these recommendations tend to be interpreted and implemented in different ways, resulting in inconsistencies even within a single text transcribed by the same expert.

Two other corpora are built on the data from personal communication channels: manually normalized WhatsApp messages (WUS) (Stark, Ueberwasser, and Göhring, 2014; Ueberwasser and Stark, 2017) and SMS messages (SMS) (Stark, Ueberwasser, and Ruef, 2009-2015). The last two resources address additional writing variation which is due to CMC factor, such as vowel reduplication and unconventional abbreviations.

Text normalization methods are not only useful for developing downstream applications. They allow to automatically add normalization layer of annotation to a corpus of transcribed spoken language. Such resources serve as a precious material for dialectological and theoretical linguistic research. For example, Scherrer, Samardžić, and Glaser (2019) propose methods to extract dialectal variation patterns from normalized data which they test on ArchiMob corpus. Stark and Meier (2017) use WUS corpus to investigate the argument drop and the causes of such language phenomena in messaging.

2.2.3 Inflection Generation

Languages with rich inflection morphology are challenging for NLP downstream applications due to data sparsity. The more word forms one particular lexeme has, the more unlikely it is to observe all of them in a single corpus. One way to reduce data sparsity is to explicitly model a mapping from an inflected form to its lemma and a sequence of inflection feature values the form encodes, and back. For example, in machine translation domain, Fraser et al. (2012) reduce diverse inflected forms in the target language into the corresponding base forms, or lemmas. At test time, they predict an abstract inflection tag for each translated lemma, which is then transformed into a proper word-form. They rely on hand-crafted morphological generators which is a traditional way to model inflection morphology.

In a more recent work, Tan et al. (2020) apply the same idea to self-supervised language models (Devlin et al., 2019): they reduce word forms to lemmas before applying subword tokenization and reinjecting the inflection information into the encoded sequence as special symbols. They show that modelling inflection explicitly helps self-suprevised language models to a) generalize to English dialects unseen during training, and b) converge faster for the machine translation task when translating from English. In their work, inflection modelling is only applied to English using a neural morphological analyzer developed specifically for English. On the other hand, "cross-linguistically, inflectional morphology exhibits a spectacular range of variation with regard to the internal structure of individual words"(Ackerman, Blevins, and Malouf, 2009). Languages differ with respect to the sizes as well as the particular values of inflection features they distinguish and the morphological processes by which these properties are encoded. Such cross-lingual variation makes the development of natural language processing (NLP) applications challenging. Therefore, we require truly language-independent models developed for languages other than English (Bender, 2011).

In this thesis, we concentrate on learning inflection in a) a supervised setting and b) using publicly available crowdsourced data - wiktionary - for inflection tables across many languages. Specifically, the task of inflection generation is to learn a generation of inflected form from a lemma and a set of inflection features' values that are encoded in the desired target form. Following example of Russian paradigm in (6), one data entry for the task is illustrated in in the following example:

(14) Inflection Generation Task

Lemma		Target Inflection Tags		Inflected Form
ruka	+	N;Sg;Gen	▷	*ruki*

where the lemma *ruka* 'hand' along with a sequence of part-of-speech and inflection features (tags) *N;Sg;Gen* is mapped to the inflected form *ruki*.

The training set may include only triples from partial paradigms. This formalization of inflection generation was proposed as a part of the SIGMORPHON shared task challenge (Cotterell et al., 2016). Inflection generation is a generalization of a paradigm completion task : the generation of a complete inflectional paradigm from a lemma, based on training data of complete paradigms, like the ones in the Ex. 6. Paradigm completion task in a supervised setting with data extracted from wiktionary was first proposed by Nicolai, Cherry, and Kondrak (2015). Inflection generation in the form (14) is more challenging than paradigm completion: the task is to learn an inflection model that not only generalizes to unseen lemmas, but also to unseen target forms of observed lemmas.

From the theoretical linguistic prospective, while paradigm completion task mimics a common task in second language (L2) pedagogy, inflection generation task is more realistic learning setting for first language (L1) acquisition. The inflection generation task is an instance of the paradigm cell filling problem (PCFG) in theoretical morphology formulated by Ackerman, Blevins, and Malouf (2009): given prior exposure to at most a subset of word forms, how does a speaker produce or interpret a novel word form of a lexeme? The problem goes back to the 'wug test' of Berko (1958) developed to test child and human knowledge of English nominal morphology. For theoretical morphologists, the difficulty of the PCFP on average is a measure of the learnability of a morphological system, with implications for morphological complexity: Ackerman and Malouf (2013) operationalize the complexity in terms of conditional entropy estimated directly from the word-form frequencies observed in a set of paradigms. Cotterell et al. (2019) develops this idea further by estimating conditional probabilities using a neural model trained on the inflection generation task.

Apart from addressing data sparsity in NLP applications on a multilingual scale and providing data for testing theories in theoretical morphology, the task of inflection generation provides an opportunity to scale up and aid theoretical linguistic research in yet a different way. As we show in Chapter 6 crowdsourced nature for data as well as high performance of neural models on the task permits to study and infer morphological patterns cross-linguistically, as long as the neural learning process is interpetable.

Chapter 3

Methodological Background

When dealing with language data, it is very common to work with sequences, such as sequences of symbols (forming subwords or words), sequences of words, or sequences of sentences. Language modelling is a fundamental NLP task of assigning probabilities to strings of text in a given language. Language model (LM) is a crucial component in downstream applications which model sequence transformations. For example, machine translation (MT) models translation of sentences, i.e. sequences of words (and ultimately, symbols), from one language to another. MT problem gave rise to two popular frameworks for modelling sequence transformation, statistical machine translation (SMT) and neural encoder-decoder model (ED). However, LM is integrated differently into each of this model.

SMT and Statistical Language Model For decades, SMT paradigm was a dominant approach to model sequence transformation. It was originally introduced by Weaver (1955) and later revived in seminal work of Brown et al. (1993). SMT framework applies a noisy-channel model (Shannon and Weaver, 1949). A noisy channel model conceptualizes communication as a problem of reconstructing (decoding) a message which is transmitted (encoded into a noisy version) through a noisy communication channel. SMT models translation as uncovering the intended message (i.e. the translation into target language) where the message is encoded by the source language.

One of the crucial components in SMT framework is the language model. More specifically, there are two main components in this framework: a translation model generates a hypothesis, i.e. output strings possibilities given an input string, while the language model assigns probabilistic scores to the hypothesis. Model components in SMT are usually build using count-based methods. Estimated in such a way language model is referred to as a statistical language model.

ED and Neural Language Model Neural network approach to sequence transformation, known as encoder-decoder, was first proposed by Cho et al. (2014b), and shortly after by Sutskever, Vinyals, and Le (2014) who independently developed this

architecture and were first to obtain state-of-the-art translation results with this approach. Yet again, the language model is a crucial component of this sequence transformation framework. However, in this approach, the language model is estimated using neural networks, namely recurrent neural networks (RNN). Encoder-decoder system comprises two RNN language models: one of them encodes a source sequence into a vector representation while the other generates (decodes) an output sequence.

Later, the ED model was improved by introducing the attention mechanism, originally proposed by Bahdanau, Cho, and Bengio (2015) and later generalized and simplified by Luong, Pham, and Manning (2015). Attention mechanism dynamically integrates the signal from different parts of an input sequence and serves as an explanation of ED systems by answering the question: which parts of the input were most important for generating particular part of the output? In this way, integrating the attention component led to more expressive and interpretable ED models.

Computational considerations for training models with RNNs led to the development of the new generation of ED models, named Transformers (Vaswani et al., 2017). In particular, the inherently sequential nature of recurrent networks inhibits the use of parallel computational resources. To overcome this challenge, Transformers replace RNNs in ED models with feed-forward neural networks. However, such replacement results in a significant problem: feed-forward networks model inputs and output of a fixed length, whereas sequence transformation problems are best expressed with sequences whose lengths are not known a-priori. To address these problems, Transformers make use of multihead self-attention mechanism, which is the key innovation of this architecture. Transformers gave rise to neural language models which use self-attention mechanism along with self-supervision training (Devlin et al., 2019). Such approach to language modelling, named contextual language models, currently dominate multiple key NLP benchmarks.

Multi-level ED and LM for Upstream Processing Encoder-decoder model was a significant breakthrough in NLP: after its introduction to machine translation, it was easily adapted to other downstream applications which can be cast as sequence transformation, e.g. text summarization and question answering. Upstream processing methods which we consider in this thesis can be cast as sequence transformation too. Compared to the setup of the downstream applications where ED framework was initially applied on a word, and later on a subword level, sequence transduction in upstream processing is modelled on a character-level. Therefore, many of the upstream tasks were previously well solved first with SMT, then with ED framework by applying such models on a character-level.

We argue that there is a strong linguistic motivation to integrate a higher-level signal into the character-level framework when solving three upstream tasks we consider in this thesis (Section 2). We take the character-level encoder-decoder model (cED) as a starting point and integrate components which operate on higher-level units. We build these components using a) two approaches to LM, statistical language model

and RNN language model; b) self-attention mechanism. Such integration is challenging due to a mismatch in atomic units. In our models, we address this challenge by proposing solutions which integrate such components separately on the encoder and decoder side of the cED framework.

Outline This chapter gives a formal description of the models that will be used as basic building blocks for developing our multi-level approaches in later chapters. We mostly follow the descriptions given in Goldberg (2017) but adapt the notations.

The rest of this chapter is organized as follows. In Section 3.1, we describe a probabilistic framework for language modelling. We then discuss how to estimate the parameters of statistical language models in Section 3.1.1. In Section 3.1.2, we first give background information on neural networks: we describe the most typical neural network functions and approaches to estimate their parameters. We then use this terminology to introduce RNN functions and how they are applied to language modelling. We build upon the RNN language model description when introducing the RNN encoder-decoder model with attention mechanism in the next subsection. Finally, we present a generalized formalization of the attention mechanism in Section 3.3, in order to introduce self-attention network.

3.1 Language Model (LM): String Scoring and Generation

The goal of language modelling task is to learn a probability distribution over sequences of symbols pertaining to a language. Formally, given a vocabulary set Σ of a language L, all possible sequences that can be constructed from vocabulary units $x \in \Sigma$ form a set Σ^*. Language L is therefore a subset of a set of all possible sequence: $L \subseteq \Sigma^*$. In this formulation, we abstract from vocabulary units, which can be single characters or sequences of characters (i.e. subwords or words in language L). Given a string $x_{1:n} = x_1 \dots x_n \in \Sigma^*$, language model assigns a probability to this string $P(x_{1:n})$, thereby measuring how probable for this string to be generated from the language L.

Using the chain-rule, probability of a string $x_{1:n}$ can be expanded as:

$$P(x_{1:n}) = P(x_1)P(x_2|x_1)P(x_3|x_{1:2}) \dots P(x_n|x_{1:n-1}) \tag{3.1}$$

To build a LM, one has to estimate conditional probabilities on the right-hand side. Once estimated, conditional probabilities can be directly used for string scoring, i.e. assigning a probability to a string by Eq. 3.1. Alternatively, conditional probabilities can be used to generate next most probable element (vocabulary unit) of a string, i.e. next most probable element of a string $x_{1:n-1}$ is a vocabulary unit $x_n \in \Sigma$ maximizing probability $P(x_n|x_{1:n-1})$.

In the next two sections, we will describe two approaches to model conditional probabilities (Eq. 3.1): count-based (Section 3.1.1) and neural network-based (Section 3.1.2). In both models, for the purpose of efficient parameter estimation and string generation, vocabulary Σ is often reduced to a subset $V \subseteq \Sigma$, by choosing m most frequent vocabulary units observed in language data. The vocabulary set V is extended to include the unique symbols *<unk>* for unknown units, *<s>* for sequence start marking, and *</s>* for end-of-sequence marking. Each data sequence is padded with the beginning-of-sequence and end-of-sequence symbols. When LM is used for a sequence generation, generation stops when end-of-sequence symbol is produced. The unknown symbol *<unk>* is used to replace all out-of-vocabulary (OOV) symbols after reducing Σ to V.

3.1.1 Statistical LM

In statistical language models, conditional probabilities in Eq. 3.1 are estimated with a count-based approach. For example, in order to estimate probability that a string $x_{1:n-1}$ has a continuation x_n:

$$P(x_n|x_{1:n-1}) = \frac{P(x_{1:n})}{P(x_{1:n-1})} \tag{3.2}$$

we first collect the counts $\#\{x_{1:n-1}\}$ and $\#\{x_{1:n}\}$ of how many times strings $x_{1:n-1}$ and $x_{1:n}$ appear in a big corpus of text. These counts are then used to estimate the probabilities in Eq. 3.2 as follows:

$$P(x_n|x_{1:n-1}) = \frac{P(x_{1:n})}{P(x_{1:n-1})} \approx \frac{\#\{x_{1:n}\}}{\#\{x_{1:n-1}\}} \tag{3.3}$$

Such approach, however, has a major drawback: not all possible strings $x_{1:n}$ will be observed, especially when n is high, resulting in zero probabilities. To address this problem of data sparsity, a simplifying markov assumption is made. Specifically, we fix a parameter k and assume that the probability of a next element in a sequence depend only on k previous elements, i.e.

$$P(x_n|x_{1:n-1}) \approx P(x_n|x_{n-k:n-1}) \tag{3.4}$$

The resulting model is referred to as a k-th order language model.

While markov assumption eliminates the problem of absence of longer sequence in a corpus, it does not fully solve the problem of zero probabilities in shorter sequences. There are still chances that some of the sequences of a length up to $k+1$ are probable but unobserved in the corpus. Such sequences would be assigned a zero probability by LM. To address this issue, many elaborated smoothing techniques have been proposed. Smoothing techniques aim to redistribute probability mass from lower-order sequences of smaller length to unobserved higher-order sequences of greater length. The current state-of-the-art statistical language modeling technique uses modified Kneser Ney

smoothing (Chen and Goodman, 1996), which is a variant of the technique proposed by Kneser and Ney (1995). For an overview of smoothing techniques, we refer to Chen and Goodman (1996) and Goodman (2001).

Statistical language models are easy to train, scale to large corpora, and work well in practice. They do, however, have an important shortcoming: markov assumption prevents them from capturing long-range dependencies. Language modelling with Recurrent Neural Networks (RNN), introduced in the next section, provides an alternative framework which allows to relax markovian assumption.

3.1.2 RNN LM

The Recurrent Neural Network (RNN) (Elman, 1990) is a type of neural network functions which are particular flexible in modelling sequential inputs. Language modelling with RNN relaxes markov assumption in k-th order statistical language models and allows for conditioning on the entire history, i.e. all proceeding vocabulary units in a sequence. This provides a very powerful framework for capturing statistical regularities in sequential inputs, including long-range dependencies.

In the following first two subsections we set the notations and terminology for neural network modelling, in order to introduce neural architectures in this and later chapters. We will first introduce the most common types of neural network functions. We then discuss how parameters in neural architectures are estimated in a supervised machine learning framework. In the next two subsections, we will proceed with description of the RNN functions and how they are applied to language modelling.

Neural Network Functions

The simplest type of neural network functions — *feed-forward networks* — can be represented as a stack of high-dimensional linear functions separated by nonlinear functions. High-dimensional *linear* function describes transformation between two vector spaces as:

$$\begin{gathered} f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b} \\ \mathbf{x} \in \mathbb{R}^{d_{in}}, \quad \mathbf{W} \in \mathbb{R}^{d_{out} \times d_{in}}, \quad \mathbf{b} \in \mathbb{R}^{d_{out}} \end{gathered} \tag{3.5}$$

where input vector $\mathbf{x}$ from d_{in}-dimensional space $\mathbb{R}^{d_{in}}$ is mapped to a vector $f(\mathbf{x})$ in d_{out}-dimensional space $\mathbb{R}^{d_{out}}$. The transformation is described by the matrix $\mathbf{W}$ and vector $\mathbf{b}$, usually referred to as a bias term.

The most popular non-linear functions, often referred to as activations, include the *sigmoid* function:

$$\sigma(x) = 1/(1 + e^{-x}) \tag{3.6}$$

and the *tanh* function:

$$\tanh(x) = (e^{2x} - 1)/(e^{2x} + 1) \tag{3.7}$$

We illustrate feed-forward network using a common class of functions in this class, called the multi-layer perceptron (MLP). The MLP with one hidden layer is

formalized as:

$$\begin{gathered} NN_{MLP}(\mathbf{x}) = \mathbf{W}^2 g(\mathbf{W}^1\mathbf{x} + \mathbf{b}^1) + \mathbf{b}^2 \\ \mathbf{x} \in \mathbb{R}^{d_{in}}, \quad \mathbf{W}^1 \in \mathbb{R}^{d_1 \times d_{in}}, \quad \mathbf{b}^1 \in \mathbb{R}^{d_1}, \quad \mathbf{W}^2 \in \mathbb{R}^{d_{out} \times d_1}, \quad \mathbf{b}^2 \in \mathbb{R}^{d_{out}} \end{gathered} \tag{3.8}$$

It describes transformation of the input vector $\mathbf{x}$ from d_{in}-dimensional space to the d_{out}-dimensional output vector. Input and output vectors are referred to as input and output layers, correspondingly. The mapping is performed by first applying a linear transformation, corresponding to matrix $\mathbf{W}^1$ and bias term $\mathbf{b}^1$, resulting in a first linear layer $\mathbf{W}^1\mathbf{x} + \mathbf{b}^1$. Then, a non-linear function g is applied element-wise to each of the resulting d_1 dimensions resulting in a vector called hidden layer. Finally, this vector is transformed by the second linear layer, described by the matrix $\mathbf{W}^2$ and the bias term $\mathbf{b}^2$, into the d_{out}-dimensional output vector. One can add more hidden layers to the described architecture, i.e. more transformations of the form $g(\mathbf{W}^1\mathbf{x} + \mathbf{b}^1)$, resulting in more deep architectures. In this thesis, we will denote by MLP the architecture with one hidden layer as in Eq. 3.8.[1]

Neural network functions provide a flexible framework for modelling classification problems which often appear in NLP tasks. For example, in language modelling, the task of predicting next element in a sequence can be seen as a multi-class classifier. More specifically, given an already generated prefix $x_{1:k-1}$, it predicts the next element x_k in the string by modelling a probability distribution over the m units in vocabulary which can be seen as m classes. This probability distribution indicates how probable it is for the next unit x_k to belong to one of the m classes.

To model m-class classification with neural networks, the prediction of the network (the output layer) has to be a m-dimensional vector, e.g. $d_{out} = m$. It is then transformed into a probability distribution through a `softmax` function:

$$\text{softmax}(\mathbf{x}) = \left[\frac{e^{x_1}}{\sum_i e^{x_i}}, \ldots, \frac{e^{x_m}}{\sum_i e^{x_i}}\right], \quad \mathbf{x} \in \mathbb{R}^m \tag{3.9}$$

This results in an output vector with positive entries summing to 1, which can be therefore interpreted as a distribution over class assignments.

Parameter Estimation in Neural Network Models

The parameters of neural networks are usually estimated in a supervised machine learning setting. We will refer to the network parameters, i.e. the matrices and the bias terms that define the linear transformations, as Θ.

[1] The nonlinear activation functions have a crucial role in the neural network's ability to represent complex functions. It was shown by Hornik, Stinchcombe, and White (1989) and Cybenko (1989) that MLP is a universal approximator — it can approximate with any desired non-zero amount of error a family of functions that includes all continuous functions on a closed and bounded subset of $\mathbb{R}^n$, and any function mapping from any finite dimensional discrete space to another. In practise, however, more complex architectures are used: while this theoretical result states that a representation exists, it does not provide any guidance on how to set the parameters based on training data.

In a supervised setting, we are faced with a dataset of n input examples $x_1, \dots x_n$ and their corresponding labels $y_1, \dots, y_n$. Our goal is to produce a function f that correctly maps inputs x_i to outputs y_i, and generalizes well to new examples. The set of example which is used to infer the function is referred to as training data, while the process of finding this function and setting its parameters as training. In machine learning framework, finding a model which generalizes well is achieved with three components: loss function, learning (or, training) algorithm and three-way dataset split.

Loss function Loss function quantifies how far the predictions of a model are from true prediction on a set of training examples. Formally, a loss function L assigns a numerical score (a scalar) to a predicted output $\hat{y}_i$ given the true expected output y_i. The choice of a loss function is specific to a task being solved.

Training algorithm The parameters Θ of a function f are set (or, trained) by a training algorithm. The goal of the training algorithm is to set the values of the parameters in order to minimize the loss L over the training examples. More formally, given a labeled training set (x_i, y_i), $i = 1, \dots, n$, a per-instance loss function L and a parameterized function $f(\mathbf{x}; \Theta)$, we define the corpus-wide loss with respect to the parameters, as the average loss over all training examples:

$$\mathcal{L}(\Theta) = \frac{1}{n} \sum_{(x_i, y_i)} L(f(\mathbf{x}_i; \Theta), y_i) \tag{3.10}$$

The parameters Θ are then set such that the value of $\mathcal{L}$ is minimized:

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}}\, \mathcal{L}(\Theta) \tag{3.11}$$

A common solution to optimization problem 3.11 is to use gradient-based methods: such techniques iteratively search for parameters which minimize a loss function. More concretely, gradient-based methods work by repeatedly computing an estimate of the loss $\mathcal{L}$ over the training set, computing the gradients of the parameters, with respect to the loss estimate, and moving the parameters in the opposite directions of the gradient. The different optimization methods differ in how the error estimate is computed, and how 'moving in the opposite direction of the gradient' is defined.

Three-way dataset split The three-way dataset split is an established way in machine learning to measure generalization. Concretely, the dataset is split into into train, validation (also called development), and test sets. The parameters are estimated using train data, whereas model selection should be performed based on the validation set. Besides selecting a type of a model, model selection includes setting hyperparameters of a single model, i.e. parameters which values have to be set manually. Then, a single run of the final model over the test set will give a good

estimate of its expected quality on unseen examples. To measure how well the model maps inputs to outputs on the test dataset, a task-specific evaluation metric is usually applied.[2]

RNN

The RNN is a natural generalization of the feed-forward networks to sequences. On a high-level, RNN is a function which takes as an input an arbitrary length sequence of d_{in}-dimensional vectors $\mathbf{x}_{1:n} = \mathbf{x}_1 \dots \mathbf{x}_n$, $\mathbf{x}_i \in \mathbb{R}^{d_{in}}$ and returns as an output a sequence of n d_{out}-dimensional vectors $\mathbf{h}_{1:n}$, $\mathbf{h}_i \in \mathbb{R}^{d_{out}}$ (commonly referred to as RNN states). More formally, the RNN is defined recursively, by means of neural network functions R, O and state vectors $\mathbf{s}_i$, $i = 1, \dots, n$, $\mathbf{s}_i \in \mathbb{R}^{f(d_{out})}$:

$$\begin{aligned} \mathbf{h}_i &= O(\mathbf{s}_i) \\ \mathbf{s}_i &= R(\mathbf{s}_{i-1}, \mathbf{x}_i) \end{aligned} \tag{3.12}$$

where at step i of the recursion, function R takes as an input a state vector $\mathbf{s}_{i-1}$ and an input vector $\mathbf{x}_i$ and returns a new state vector $\mathbf{s}_i$. The state vector $\mathbf{s}_i$ is then mapped to an output vector $\mathbf{h}_i$ using function O. The base of the recursion is an initial state vector, $\mathbf{s}_0$, which is also an input to the RNN. The functions R and O are the same across the sequence positions, but the RNN keeps track of the states of computation through the state vector $\mathbf{s}_i$ that is being passed across invocations of R.

The recursive definition in Eq. 3.12 implicitly defines an output vector $\mathbf{h}_i$ for each prefix $\mathbf{x}_{1:i}$ of the sequence $\mathbf{x}_{1:n}$. We will use star notation RNN* to denote a function which returns full sequence and keep RNN notation to denote only one step in the recurrence:

$$\text{RNN}^*(\mathbf{x}_{1:n}) = \mathbf{h}_{1:n} \tag{3.13}$$

$$\text{RNN}(\mathbf{x}_{1:n}) = \mathbf{h}_n \tag{3.14}$$

The functions R and O can be instantiated differently, resulting in different RNN types. A particular popular class of RNNs is gating-based RNN architectures. They are designed to tackle a problem of vanishing gradients: gradients become exceedingly close to 0 in deep neural networks, especially so in recurrent networks (Pascanu, Mikolov, and Bengio, 2013). In this thesis, we will use gated RNN architecture called Long Short-Term Memory (LSTM), proposed by Hochreiter and Schmidhuber (1997).[3]

[2] Often, such metric is different from the loss function. For example, one can choose accuracy of the model predictions on the test set. Such metrics, while reflecting well the goal of a task, are often not differentiable and therefore, cannot be chosen as loss function. This results in a known problem of loss-evaluation mismatch. While this problem is being increasingly addressed and new training methods are proposed (e.g. in Ranzato et al. (2016) and Edunov et al. (2018), among others), we do not attempt to tackle it in the models developed in this thesis.

[3] Using the O function in RNN definition is somewhat non-standard. We follow the definition from Goldberg (2017), where it is introduced in order to unify the different RNN models. In some RNN

The RNN serves as a trainable component of a larger network which makes a final prediction on a concrete task. Used in this way, the RNN learns to encode properties of the input sequences that are useful for the further prediction task. Parameters of both RNN and the larger network are then optimized with respect to a loss on the final predictions. In the the next subsection, we will introduce an example of a larger network when discussing how RNN can be applied to language modelling task.

RNN LM

To apply RNN abstraction to language modelling, we first introduce an embedding layer (or, lookup layer) in order to turn sequences of vocabulary units $x_{1:n}$ into sequences of d_{in}-dimensional vectors $\mathbf{x}_{1:n}$, $\mathbf{x}_i \in \mathbb{R}^{d_{in}}$. The parameters of an embedding layer constitute a matrix $\mathbf{E} \in \mathbb{R}^{d_{in} \times |V|}$ where each row corresponds to a different unit in the vocabulary V. The lookup operation is then simply indexing: if x_i is a vocabulary unit indexed with an index k, its corresponding vector is a column k in the matrix $\mathbf{E}$.

RNN language model (RNN LM) is constructed as a transducer which sequentially predicts an output symbol for each input symbol. More specifically, to generate the next symbol in a sequence, at each prediction step $i+1$ transducer's input is tied to the output at position i. Formally, at each step, a conditional distribution over the next element in a sequence is modelled as:

$$\begin{aligned} P(x_{i+1}|x_{1:i}) &= \text{softmax}(\mathbf{W}O(\mathbf{s}_{i+1}) + \mathbf{b}) \\ \mathbf{s}_{i+1} &= R(\mathbf{s}_i, \hat{\mathbf{x}}_i) \\ \hat{x}_i &\sim P(x_i|x_{1:i-1}) \end{aligned} \tag{3.15}$$

where after predicting a distribution over the previous output symbol $P(x_i|x_{1:i-1})$, a symbol $\hat{x}_i$ is chosen and the corresponding embedding vector $\hat{\mathbf{x}}_i$ is fed as the input to the next step. The linear layer, corresponding to the matrix $\mathbf{W} \in \mathbb{R}^{|V| \times d_{out}}$ and bias $\mathbf{b} \in \mathbb{R}^{|V|}$, followed by a `softmax` function (Eq. 3.9), transforms the hidden state $O(\mathbf{s}_{i+1})$ into a distribution over vocabulary items in V. This transformation is referred to as an output layer. The distribution obtained at the output layer is used to predict the next output symbol x_{i+1}.

The model is trained using cross-entropy loss function which quantifies how far a predicted distribution is from the true one. Let $P = p_1, \dots p_m$ and $Q = q_1, \dots q_m$ be a predicted and true distribution over classes (i.e. vocabulary units) $1, \dots, m$ correspondingly. The cross-entropy between distributions Q and P is defined as:

$$\text{Cross-Entropy}(Q, P) = -\sum_{i=1}^{m} q_i \log(p_i) \tag{3.16}$$

models, O is the identity mapping (e.g. Elman RNN (Elman, 1990) and the GRU architectures (Cho et al., 2014b), whereas in the LSTM architecture O selects a fixed subset of the state.

This equation simplifies for RNN LM at each prediction step to:

$$\text{Cross-Entropy}(Q, P) = -\log(p_k) \tag{3.17}$$

where k is the correct class assignment.

For the purpose of parameters training, cross-entropy loss is aggregated over the training examples, i.e. sequences $x_{1:n}$, as:

$$\mathcal{L}^{CE}(\Theta) = -\sum_{x_{1:n}} \sum_{i} \log(P(x_i|x_{1:i-1})) \tag{3.18}$$

As explained earlier in this section, the parameters are set by minimizing this loss function with gradient-based methods.

3.2 Encoder-Decoder: Conditioned Generation with RNN LM

Flexibility of RNN LM architecture described in Eq. 3.15 permits an extension to conditioned generation framework. This framework allows generation of sequences where probability of the next element in a sequence can be conditioned on a generalized context. The context represents any context information deemed to be useful for a specific task.

In a conditioned generation framework, conditional probability distribution at each step in a sequence is modelled as:

$$\begin{aligned} P(x_{i+1}|x_{1:i}, \mathbf{c}) &= \text{softmax}(\mathbf{W}[O(\mathbf{s}_{i+1}); \mathbf{c}] + \mathbf{b}) \\ \mathbf{s}_{i+1} &= R(\mathbf{s}_i, \hat{\mathbf{x}}_i) \\ \hat{x}_i &\sim P(x_i|x_{1:i-1}, \mathbf{c}) \end{aligned} \tag{3.19}$$

where prediction layer receives as an input a vector obtained by a concatenation of a) RNN output vector; and b) a context vector $\mathbf{c}$.

The conditioned generation formulation gave a rise to RNN-based framework for modelling transduction from a variable-length input sequence to a variable-length output sequence, also named as sequence-to-sequence or encoder-decoder (ED). In such framework, one RNN is used to encode input sequence (encoder network), another RNN is used to generate output input sequence (decoder network). The context vector $\mathbf{c}$ aims to summarize information from the encoded input sequence which is relevant for producing the output sequence. For example, in the initially proposed ED models (Cho et al., 2014b; Sutskever, Vinyals, and Le, 2014), context vector $\mathbf{c}$ is taken as the last state of the encoder RNN. This results in a rather strong requirement for a model to summarize all the information from the input sequence required for generation in a fixed-length vector. The encoder-decoder system with an attention network (Bahdanau, Cho, and Bengio, 2015) relaxes the condition that the entire

source sequence is encoded as a single vector. Instead, the decoder uses an attention module in order to decide on which parts of the encoding input it should focus. In the following, we will formulate encoder-decoder system with attention which serves as a starting point for developing multi-level subword processing models in this thesis. The exact formulations largely follow the encoder-decoder system described in Luong, Pham, and Manning (2015).

In the sequence transduction task, the goal is to learn a mapping from a source sequence $x = x_{1:n}$ to a target sequence $y = y_{1:n}$. In order to formalize the task, we define two vocabulary sets, Σ_x and Σ_y, consisting of the vocabulary units that form the input (source) and output (target) sequences correspondingly. Then, the sequence transduction task is to generate a target sequence form $y \in \Sigma_y^*$, given an input sequence $x \in \Sigma_x^*$. For example, the task of morphological segmentation (Eq. 12) can be seen as a sequence transduction task where the source sequence is an input word and the target is its segmentation. The vocabulary set Σ_x (Σ_y) can be chosen as characters of the source (target) sequences. Following the vocabulary notations introduced in Section 3.1, input and output vocabularies are reduced to sets of the most frequent items, V_x and V_y correspondingly, and extended to include sequence padding symbols and OOV symbol. The input and output sequences in the data are padded with the padding symbols.

The model transforms the input sequence $x = x_{1:n}$ into a sequence of hidden states $\mathbf{h}^x = \mathbf{h}^x_{1:n}$ using a bidirectional RNN encoder (Graves, 2008; Schuster and Paliwal, 1997). In such architecture, two RNNs are used: forward RNN (denoted in the following as $\overrightarrow{f}$) reads the input sequence $x_{1:n}$ as it is, from left to right, while the second RNN (denoted as $\overleftarrow{f}$) reads the input sequence in a reverse order. The state representation $\mathbf{h}^x_i$ of the input symbol at position i is composed by a concatenation of the forward and backward RNN output vectors at this position:

$$\mathbf{h}^x_i = [\overrightarrow{f}(\mathbf{x}_{1:i}); \overleftarrow{f}(\mathbf{x}_{n:i})] \tag{3.20}$$

Much like the RNN relaxes the Markov assumption and allows looking arbitrarily back into the past, the bidirectional RNN relaxes this requirement further, allowing to look arbitrarily far at both the past and the future within the sequence.

The decoder RNN generates a variable length output sequence $y = y_{1:m}$ given the internal input representation $\mathbf{h}^x$. At each prediction step t, the conditional probability over output vocabulary units is modeled as a function of the current decoder hidden state $\mathbf{s}_t$ and the current context vector $\mathbf{c}_t$:

$$\begin{aligned} P(y_{t+1}|y_{1:t}, x) &= \text{softmax}(\mathbf{W}_o\tilde{\mathbf{s}}_{t+1} + \mathbf{b}_o) \\ \tilde{\mathbf{s}}_{t+1} &= \tanh(\mathbf{W}_a[O(\mathbf{s}_{t+1}); \mathbf{c}_{t+1}]) \\ \mathbf{s}_{t+1} &= R(\mathbf{s}_t, \hat{\mathbf{y}}_t) \\ \hat{y}_t &\sim P(y_t|y_{1:t-1}, x) \end{aligned} \tag{3.21}$$

The last layer, determined by matrix $\mathbf{W}_o$, bias $\mathbf{b}_o$ and `softmax` function, is usually referred as output layer. It is applied to the vector $\tilde{\mathbf{s}}_{t+1}$ which is obtained by 'updating' the decoder output vector with a context information $\mathbf{c}_t$. The vector $\tilde{\mathbf{s}}_{t+1}$ is usually referred to as an attentional decoder state.

The context vector $\mathbf{c}_t$ is computed at each step t as a weighted sum of the encoder hidden states $\mathbf{h}^x$:

$$\mathbf{c_t} = \sum_{i=1}^{n} a_t^i \mathbf{h}_i^x \tag{3.22}$$

where attention weights $\mathbf{a}_t \in \mathbb{R}^n$ are derived from a learned alignment between input and output positions. More concretely, alignments scores $\boldsymbol{\alpha}_t \in \mathbb{R}^n$ are computed by a general scoring function (Luong, Pham, and Manning, 2015) which 'compares' current decoder state $\mathbf{s}_t$ to each of the input hidden states $\mathbf{h}_i^x$, with a subsequent mapping of scores to probabilities through a `softmax` function:

$$\begin{aligned} \alpha_t^i &= \mathbf{s_t}^\intercal \mathbf{W}_a \mathbf{h}_i^x \\ \mathbf{a}_t &= \text{softmax}(\boldsymbol{\alpha}_t) \end{aligned} \tag{3.23}$$

All the components of the system - encoder, decoder and attention module - are trained jointly. The training objective is to the minimize cross-entropy loss over the training pairs (x, y):

$$\mathcal{L}^{CE}(\Theta) = -\sum_{(x,y)} \sum_{t} \log P(y_t | y_{1:t-1}, x) \tag{3.24}$$

where Θ is a set of the network's parameters.

Input-feeding mechanism. In our formulations of the conditioned generation for both RNN LM and RNN encoder-decoder, we depart from the previous formulations by not considering input-feeding mechanism. Such mechanism considers as an input to the recurrence function R a concatenation of the previous prediction ($\hat{\mathbf{x}}_i$ in Eq. 3.19 and $\hat{\mathbf{y}}_t$ in Eq. 3.21) with a vector which carries context information. For example, Goldberg (2017) use the context vector $\mathbf{c}$ and feed a concatenated vector $[\hat{\mathbf{x}}_i; \mathbf{c}]$ to the function R in RNN LM Eq. 3.19. In RNN encoder-decoder (Eq. 3.21), Bahdanau, Cho, and Bengio (2015) use a context vector, similar to our $\mathbf{c}_t$, while Luong, Pham, and Manning (2015) use a previous attentional decoder state $\tilde{\mathbf{s}}_t$. We choose a simpler architecture of feeding only the previous prediction in the recurrence in order to unify encoder-decoder systems used in this thesis. In those experiments, where we specifically apply input-feeding, we provide details of exact modifications to the basic encoded-decoder system described in this chapter.

Training versus test-time generation. The prediction $\hat{y}_t$ at each time step in Eq. 3.21 is chosen differently at training and testing time. During train time, at each prediction step, embedding of a true previous symbol y_t is fed. This training approach is often called teacher-forcing. In contrast, at test time, when generating a

sequence, the model's prediction $\hat{y}_t$ is used. The choice of the prediction depends on the decoding algorithm used. Such algorithm describes the process of finding globally most likely sequence $\hat{y}$ according to the distribution $P(y|x)$:

$$\hat{y} = \underset{y}{\operatorname{argmax}} \log P(y|x) = \underset{y}{\operatorname{argmax}} \sum_t \log P(y_t|y_{1:t-1}, x) \tag{3.25}$$

where conditional probabilities are estimated according to Eq. 3.21. In the experiments in the thesis we use beam search as a decoding strategy. We postpone its formalization till Chapter 4 where a multi-level modification to beam search is introduced.

3.3 Cross-Attention vs Self-Attention

The attention mechanism of the encoder-decoder systems, introduced in the previous section, represents an instance of a more general attention head framework. In this section, we give a formal description of the concept of an attention head module. We follow the terminology of an attention head module in the context of the transformer encoder-decoder model (Vaswani et al., 2017) but we simplify it for our needs. We will use the attention head terminology in order to distinguish between two types of the attention modules that can be used in the encoder-decoder framework: cross-attention head and self-attention head.

Attention Head Module An attention head module is designed to calculate attention vectors of the system. Formally, given an input sequence of vectors $\mathbf{H} = \mathbf{h}_1 \dots \mathbf{h}_J$, $\mathbf{h}_k \in \mathbb{R}^{D1}$ and a query vector $\mathbf{q} \in \mathbb{R}^{D2}$, attention head module computes two components: attention weights $\mathbf{a} \in \mathbb{R}^J$ and an attention head vector $\mathbf{c} \in \mathbb{R}^{D1}$:

$$\alpha_j = \mathbf{q}^\intercal \mathbf{W}_a \mathbf{h}_j, \quad \mathbf{c} = \sum a_j \mathbf{h}_j \tag{3.26}$$

where $\mathbf{W}_a \in \mathbb{R}^{D2 \times D1}$ and attention weights $\mathbf{a} \in \mathbb{R}^J$ are obtained by a mapping function from real values to probabilities, applied to the alignment scores $\boldsymbol{\alpha} \in \mathbb{R}^J$. A common choice for such mapping function is the `softmax` (Eq. 3.9) which yields dense attention weights: all elements in the input always make at least a small contribution to the final attention head vector. In this thesis, we also consider an alternative to this function, named `sparsemax`. It produces sparse attention weights using the Euclidean projection of real values onto the simplex (Martins and Astudillo, 2016).

Hereafter, we refer to the construction of an attention head (Eq. 3.26) as scoring a sequence of vectors $\mathbf{H}$ with a query vector $\mathbf{q}$.

Self-Attention Head In the encoder-decoder system, we refer to a head attention module as a self-attention if the sequence of vectors $\mathbf{H}$ and the query vector $\mathbf{q}$ are constructed only on the input side (both $\mathbf{H}$ and $\mathbf{q}$ refer to particular RNN encoder

states), or only on the output side (both $\mathbf{H}$ and $\mathbf{q}$ refer to particular RNN decoder states).

Cross-Attention Head The attention mechanism described in Section 3.2 can be seen as an attention head module constructed at each prediction time step t by scoring an encoded input sequence $\mathbf{h}_x$ with a decoder state $\mathbf{s}_t$. We will refer to such module as cross-attention.

Chapter 4

Synchronized Decoding for Morphological Segmentation

4.1 Introduction

With the spread of natural language processing to a broader range of languages, learning *internal word structure* becomes increasingly important for developing practical applications. NLP methods developed for English traditionally used words as their atomic processing units. Portability of such applications to languages with rich morphology becomes challenging due to *data sparsity*, especially in low-resourced languages.

For example, one layer of the cross-linguistic variation in the internal word structure is how many morphemes languages tend to pack into one word. In some languages, such as English, words are relatively short and morphologically less complex, i.e. have fewer morphemes per word. In other languages, such as Chintang, words tend to be long and encapsulate rather rich structure. To illustrate, Ex. 15 shows a phrase in Chintang annotated with glosses (see Section 2.2.1 for the introduction to interlinear glossing annotation). In this example, the verb *thaptakha* consists of several morphemes expressing the imperative mode (direct commands, prohibitions, and requests) and deixis (an indication of the addressee, in this example).

(15) cuwa thaptakha!

cuwa	*thapt*	*-a*	*-khag*	*-a*
water	move.across	-IMP	-see	-IMP[2sS]

'Bring some water over here!'

(Stoll, Mazara, and Bickel, 2017)

The information expressed by a single Chintang verb requires several words in English, as it can be seen in the glosses and in the translation. This directly leads to the sparse data problem: the more morphemes are packed into words in a language, the more unseen word forms would be introduced with any new sample of text which a downstream system has to process. To reduce data sparsity, an explicit analysis of

internal word structure, usually termed *morphological segmentation*, is often used as a preprocessing step in the downstream application.

Morphological segmentation is often performed in the form of *surface segmentation*. This method segments word forms into their constituent morfs, i.e. the word *recognizing* in English would be analyzed as *recogniz+ing*. Even more popular are unsupervised methods for surface segmentation which search for frequent subwords in data. For example, such method could produce a segmentation of a form *re+cogn+izing* for the previous example. Identified in this way segments are frequently seen in the data but only loosely related to a language's morphology. After prepossessing words using surface segmentation, the identified segments, instead of orthographic words, can be chosen as processing units of the NLP system. In this way, the system has more possibilities to learn morph-syntactic regularities of language structure and generalize better to unseen data.

The focus on surface segmentation is largely due to ease of model definition and implementation rather than linguistic correctness. Although surface segmentation is helpful for languages with concatenative morphology (we refer to Section 2.1 for the discussion of concatenative morphological patterns), it has certain limitations. Such methods fail to recognize *allomorphy*, i.e. that a single morpheme may have multiple surface forms (allomorphy is introduced in more detail in Section 2.1). Presence of allomorphy is a frequent cross-linguistic phenomenon. Even in languages with concatenative morphology where surface segmentation is considered helpful, allomorphy is often present due to phonological changes occurring at morphemes' boundaries. For example, English morpheme *-ed* marking present tense in verbs can have multiple surface realization as in: *-ed* (jumped) or *-d* (said) (see Ex. 12 for more examples). Although sufficient amount of data could provide evidence to a downstream system that both forms *-ed* and *-d* share the same linguistic function, identifying underlying morphemes rather than surface morphs becomes more important to reduce data sparsity in under-resourced languages. This deeper level of analysis is addressed in the task of *canonical morphological segmentation*, which is also the topic of our study.

Canonical morphological segmentation, illustrated in the next example, analyzes words as a sequence of canonical morphemes.

(16) Canonical Morphological Segmentation Task (Chintang)

Input Word		Segmentation
thaptakha	▷	*thapt|a|khag|a*

Each canonical morpheme is a form chosen to represent all surface realization of this morpheme. For instance, the canonical segment *-khag* corresponds to its surface realization *-kh* in Ex. 16.

In this study, we focus on developing a solution for canonical morphological segmentation which relies on the relatively small, manually analyzed corpora increasingly available in the linguistic community. In such setting, we address segmentation as a

prepossessing step for low-resource languages. Our approach has implications for theoretical linguistic research too. It allows to automatically produce annotations such as interlinear glossing annotations (Ex. 15) using small manually-build resources.

In contrast to much of the prior work on surface segmentation, we focus on supervised setting, i.e., we provide the model with gold segmentations during training time. To address our goals, we cast the task as neural sequence-to-sequence learning and take a character-level encoder-decoder model (cED) introduced in Section 3.2 as a starting point. Such model only takes into account local character transformation. We argue that a higher-level signal which can be extracted from canonical segment sequences is useful for the task too. In order to extract and integrate such higher-level signal, we develop a multi-level extension to cED system which we term "synchronized decoding".

The chapter is organised in the following way. We start by formalizing the task of word segmentation as a sequence transduction task in Section 4.2. This allows solving the task using the encoder-decoder paradigm on a character-level. We then show how a higher-level signal from morpheme sequences can be integrated into such system using our novel synchronized decoding algorithm. We relate our synchronized decoding method to the previous work and review approaches to segmentation in Section 4.3. In our experimental study, we test our algorithm's generalization ability to process unseen words by considering different data availability settings. In the first experiment, Section 4.4, we examine the performance impact of our multi-level modification to encoder-decoder on the task of canonical segmentation (Ex. 16). We complement the study with quantitative analysis on how our synchronized decoding targets segmentation of different categories of unseen words. In the second experiment, we demonstrate (Section 4.5) that further performance improvements can be achieved when running the synchronized decoding in a semi-supervised mode: in addition to parallel data to the task, we show how extra 'out-of-domain' language data can be utilized to improve performance on the task.

4.2 Synchronized Decoding Algorithm: cED+HLLM

Given an input sequence, such as the Chintang verb in Ex. 16, the task of canonical segmentation is to produce a segmented form, where we recognize that the sequence *kh* in the surface form is an instance of the light verb *khag*. The task can be formalized as a sequence transduction problem. Formally, we define two discrete alphabets, Σ of the surface symbols and Σ_{can} of the canonical symbols. For many languages these two alphabets coincide, for example in the case of English they consist of 26 letters of the Latin alphabet. In the case of Chintang, these alphabets are different: the surface symbols express more specific pronunciation features. Our task is to learn a mapping from a surface word form $x \in \Sigma^*$ (e.g., x='thaptakha'), to its canonical segmentation $y \in \Omega^*$ (e.g., y=thapt|a|khag|a). We define $\Omega = \Sigma_{can} \cup \{|\}$, where the symbol '|' marks segmentation boundaries.

To learn the mappings, we combine the general sequence transformation framework introduced in Section 3.2 — LSTM encoder-decoder with attention mechanism — trained on a character level with a language model trained on morphemes. Note that the general character-level framework already implicitly contains a kind of a language model over characters as a part of the decoder. The language model in our approach is trained over sequences of segments, providing additional information which can be extracted from character sequences. However, the mismatch in units of two systems, characters versus segments, poses a challenge for its integration in the general framework.

We tackle this problem with our "synchronization" method applied at the decoding stage: the segmentation hypotheses are expanded and scored using a combination of (a) scores from a lower-level encoder-decoder model and (b) higher-level scores of the language model. The fusion of the scores is triggered only at the segmentation boundaries. In the following, we refer to the character-level encoder-decoder component as cED and to the higher-level language model as HLLM.

In this section, we first review the standard beam-search approach to decoding in encoder-decoder LSTM framework. Then, we present our multi-level modification to the beam-search, which we term "synchronized decoding", that allows integration of a higher-level language model into the encoder-decoder system.

4.2.1 Background: Decoding with Beam Search

For a trained encoder-decoder model and a given input x, a prediction is made by searching for the most likely sequence $\hat{y}$ according to the estimated conditional probabilities:

$$\hat{y} = \underset{y}{\operatorname{argmax}} \log P(y|x) = \underset{y}{\operatorname{argmax}} \sum_{t} \log P(y_t|y_{1:t-1}, x) \tag{4.1}$$

The process of finding the most likely sequence is referred to as decoding. The space of possible predictions y, called search space, grows exponentially with the length of the predicted sequence, i.e. there are $|V_y|^{t_{max}}$ possible sequences to evaluate. Here, t_{max} is the maximum length of the output sequence. Following notation introduced in Section 3.2, V_y denotes the output vocabulary which includes output symbols from the alphabet Σ_{can} (possibly limited to a number of most frequent items), OOV symbol and special symbols for beginning (<w >) and end-of-sequence (</w >) padding.

In order to control exponential growth of the search space, decoding is usually performed with a beam-search algorithm. Beam-search combines a breadth-first strategy for evaluating sequences from the search space with a specific filter which prunes the search space. Specifically, breadth-first search operates by first evaluating all possible output sequences $y_1 \in V_y$ of length 1, then it evaluates all their continuation by one symbol $y_1 y_2 \in V_y^2$, and proceeds in such fashion until the final depth t_{max} is reached. The filtering feature of beam search reduces the search space by only passing a fixed number of possible outputs from the k-th level of depth to be evaluated at the next $(k+1)$-th level, thereby discarding, i.e. pruning, the rest of all output sequences

BeamSearch: **expansion by characters** with cED score

Input : Input word x, beam size K
Output: K-best predicted segmentations y.
Initialize Hypotheses=[<w>]
while not all $h_i \in$ Hypotheses end with </w>: do
  New_Hypotheses=[]
  foreach $h_i \in$ Hypotheses **do**
    if h_i *does not end with </w>:* **then**
      Add all $h_{i+1}^1, \ldots, h_{i+1}^{|V|}$ expansions of h_i based on cED score to New_Hypotheses
    else
      Add h_i to New_Hypotheses
  Hypotheses $=K$-best$_{cED}$(New_Hypotheses)
return Hypotheses

FIGURE 4.1: Pseudocode for Beam Search Decoding Algorithm. The segmentation hypotheses are expanded by characters and pruned using cED scores.

evaluated at the k-th level. This fixed number is called beam width. We will refer to the sequences of partial outputs passed to the next level as hypotheses.

We illustrate the beam search algorithm using our segmentation task. At the first step of decoding, $t = 1$, we initialize the set of hypothesis with K top scoring first elements y_1 where K is the beam size. The scores are logarithm of learned probabilities over vocabulary items, e.g. $\log P(y_1|x)$, $y_1 \in V_y$ for the first step. At each subsequent time step t, the cED system computes a score $\log P(y_t|y_{1:t-1}, x)$ for each possible next character y_t in the vocabulary V_y as a continuation of the segmentation hypothesis from the previous step $\{(y_1 y_2 \ldots y_{t-1})^i\}$, $i = 1, \ldots, K$. Then, each possible continuation $\{(y_1 y_2 \ldots y_{t-1})^i y_t\}$, $y_t \in V_y$, $i = 1, \ldots, K$ gets a score which is a sum of cED scores for each character, that is, a sum of the scores for a hypothesis from a previous time step $(y_1 y_2 \ldots y_{t-1})^i$ and a score for the next character y_t. Thus we get a set of $|V_y| \times K$ new hypothesis of length t together with their respective scores. All these new hypotheses at the step t can then be sorted according to their respective scores, and the top K ones are selected as candidates for the expansion at the next time step. There are two possibilities for finishing the search. Either the process stops at the time step where the best scoring hypothesis is "closed", i.e. ends with </w>. Alternatively, the search finishes when all best scoring hypotheses end with </w>. The choice of the stopping criteria depends on the application: whether we require as an output the prediction $\hat{y}$, i.e. the best scoring "closed" hypothesis, or a K-best list of predictions, i.e. the best K scoring "closed" hypotheses. Fig. 4.1 illustrates beam search in a pseudocode.

4.2.2 Synchronized Decoding: Integrating a Morpheme Language Model (HLLM) into cED

To integrate a higher-level language model into the cED system, we propose a synchronized decoding algorithm. Before the integration, we assume that an cED model and a HLLM are trained separately. The cED model is trained on character sequences in a parallel corpus where the source side consists of unsegmented words and the target side consist of the canonically segmented words. It learns local character transformations and implicitly includes a LM over the target side characters through the decoder LSTM component. We augment this model with an additional HLLM, separately trained over morpheme sequences on the target side of the corpus. HLLM scores how likely a given sequence of morphemes is in a given language. Therefore, the additional HLLM brings the frequency signal from higher-level units of the target data (segments), while the cED system operates on characters. In the following, we describe to fuse the scores of both components, cED and HLLM, using the synchronized mechanism.

In our synchronized decoding approach, we can find the most probable segmentation using a beam search algorithm (Section 4.2.1) guided by "synchronized" character-level cED and morpheme-level HLLM scores. In order to guide the beam decoding with the HLLM scores we perform a "synchronization". Specifically, we run the beam search at two levels of granularity, both with the same beam width K. The lower-level beam search expands hypotheses by characters, whereas higher-level beam search expands them by segments. First, we run the beam search at the character level using cED scores until the time step s_1, where K best hypotheses $\{(y_1 y_2 \dots y_{s1})^i\}$, $y_t \in V_y$, $i = 1, \dots, K$ end with a boundary symbol. The boundary symbol can be either end-of-sequence symbol </w> or the segmentation boundary symbol '|'. The length of the produced partial outputs $s1$ is dependent on each hypothesis i but we assume it to be the same among them to ease the notations.

We refer to the point s_1 as the first synchronization step, i.e. the first step of the higher-level beam search. At this step, we re-score the normalization hypotheses with a weighted sum of the cED score and the HLLM score:

$$\begin{aligned}\log p(y_{s1}|y_1, \dots, y_{s1-1}, X) \\ &= \log p_{cED}(y_{s1}|y_1, \dots, y_{s1-1}, X) \\ &\quad + \alpha_{HLLM} \log p_{HLLM}(y_1, \dots, y_{s1-1}) \end{aligned} \tag{4.2}$$

where α_{HLLM} is a hyperparameter. In this way, $y_1, \dots, y_{s1}$ is considered a sequence of $s1$ characters by the cED system and $y_1, \dots, y_{s1-1}$ (without the last boundary symbol) is considered one morpheme by the LM.

After the first synchronization point we continue to expand the re-scored hypotheses $\{(y_1 y_2 \dots y_{s1})^i\}$ $i = 1, \dots, K$ by characters using again only cED scores. In this way, each hypothesis accumulates a score which consists of the combined score from

the previous synchronization point and character scores from the current character-level expansion. The expansion is only performed for the "open" hypothesis, i.e. sequences which do not end with the end-of-sequence symbol. For the "closed" hypotheses its expansion is taken to be itself, i.e. they are kept as it is to be passed to the next evaluation and pruning with the higher-level beam search. We continue this process of the lower-level beam search until we get to the next synchronization point $s2$ where all K continuations $\{(y_1 y_2 \dots y_{s1})^i (y_{s1+1} \dots y_{s2})^j\}$ $j = 1, \dots, K$ for each of the K hypotheses $\{(y_1 y_2 \dots y_{s1})^i\}$ $i = 1, \dots, K$ end with a boundary symbol. As a result, we get a maximum of $K \times K$ new hypotheses which we pass to the higher-level beam search. At this point, we re-score them with a weighted sum of cED and HLLM and select the top K ones. After rescoring and pruning, we continue this process again till the next synchronization point. The decoding process ends at a synchronization point where the last symbol of the best scored hypothesis (using the combined cED and HLLM score) is an end-of-sequence symbol.

The described decoding process therefore scores the segmentation hypotheses at two levels: normally working at the character level with cED scores and adding the HLLM scores only when it hits a boundary symbol. In this way, the HLLM score helps to evaluate how probable the last generated morpheme is based on the morpheme history, that is the sequence of morphemes generated at the previous synchronization time steps.

4.2.3 The Length Constraint

Language models score sequences by multiplying conditional probabilities of each sequence element using a chain rule (3.1). Due to such probabilistic nature of language models, they give higher preference to shorter sequences. This becomes an issue in the proposed fused model described above: at the synchronization points high HLLM scores tend to stop further hypothesis expansion. For example, only the first segment can be generated as a model output if it happens to be a frequent standalone word. This leads to favoring segmentation predictions where the output is shorter than the input, which is rarely plausible in segmentation. Our early experiments confirmed this intuition, therefore we consider the length constraint component, introduced below, to be an integral part of the language model inclusion and we do not report experiments without this component.

To deal with the length issue, we add a "length constraint" component LC. The LC score is based on the difference in character length between the input word and its segmentation hypothesis. To synchronize the LC score with HLLM scoring process described before we assign it only at the synchronization time steps. Therefore, the LC score, combined with the HLLM score, helps to evaluate how probable is the last generated morpheme given the sequence of morphemes generated at the previous steps.

Assume that the input word is $X = x_1 \dots x_n$ and the produced segmentation hypothesis at the first synchronization step $s1$ is $y_1 \dots y_{s1}$ where y_{s1} is a boundary

symbol. Then the LC score assigned to the morpheme $y_1 \dots y_{s1-1}$ is calculated as the negative value of the absolute difference between the morpheme length and input word length divided by the the input length: $LC(y_1 \dots y_{s1-1}) = -(|y_1 \dots y_{s1-1}| - |X|)/|X| = -|s1 - 1 - n|/n$. At the next synchronization point $s2$ the LC score is calculated using the length of the next produced segment: $LC(y_{s1+1} \dots y_{s2-1}) = -(|y_{s1+1} \dots y_{s2-1}| - |X|)/|X|$. In a general case, the LC score for the last generated segment σ_i can be expressed as

$$LC(\sigma_i) = -(|\sigma_i| - |X|)/|X| \tag{4.3}$$

Boundary symbols are excluded for the segments length calculation.

The intuition behind the LC score is that it gives a contribution to the total score of a segmentation hypothesis showing how different the length of the hypothesis, produced so far, compared with the length of the input word. The characters in the canonical segments tend to be either inserted or deleted compared to their surface form equivalents, therefore we measure LC score using an absolute difference in the length. The higher the absolute value of the difference between the input and the hypothesis, the higher the penalty.

With the inclusion of the LC score for the length control the total score of our fusion model becomes:

$$\begin{aligned} \log p(y_{s1}|y_1, \dots, y_{s1-1}, X) & \\ &= \log p_{cED}(y_{s1}|y_1, \dots, y_{s1-1}, X) \\ &+ \alpha_{HLLM} \log p_{HLLM}(y_1, \dots, y_{s1-1}) \\ &+ \alpha_{LC} LC(y_1, \dots, y_{s1-1}) \end{aligned} \tag{4.4}$$

where the weights α_{LM} and α_{LC} are optimized on a development set.

In our experiments, we use MERT optimization (Och, 2003) to set the the weights α_{LM} and α_{LC}. This is a standard optimization routine in statistical machine translation which searches for the weights of the model components by directly maximizing the performance of the system on a development set using n-best prediction lists.

The pseudocode for the syncronized decoding algorithm is illustrated in Fig. 4.2.

4.3 Related Work

We demonstrate the novelty of our approach by first comparing the architectural aspects of our methodology to the previous work. To this end, we review how combination of encoder-decoder system with a language model over a target side data is addressed in other methods. To position our method within the domain of subword segmentation, we give an overview of the previous methods applied to the task of canonical morphological segmentation.

SyncBeamSearch: **expansion by morphemes** with combined score CS (weighted cED,HLLM,LC).

Input : Input word x, beam size K
Output: Predicted segmentation h
Initialize Hypotheses = [<w>]
while not 1-best$_{CS}$(Hypotheses) is closed with </w>: do
  New_Hypotheses=[]
  foreach $h_i \in$ Hypotheses **do**
    if h_i *is not closed with </w>:* **then**
      Add all (h_{i+1}^1, ..., h_{i+1}^K from **ModifiedBeam$_{cED}$**(h_i) to New_Hypotheses)
    else
      Add h_i to New_Hypotheses
  Hypotheses = K-best$_{CS}$(New_Hypotheses)
return 1-best$_{CS}$(Hypotheses)

ModifiedBeam$_{cED}$: **expansion by characters** with cED score

Input : Partial hypothesis h for an input word x, beam size K
Output: K-best expansions of h, closed with '|' or </w>.
Initialize Hypotheses=[h]
while not all $h_i \in$ Hypotheses are closed with '|' or </w>: do
  New_Hypotheses=[]
  foreach $h_i \in$ Hypotheses **do**
    if h_i *is not closed with '|' or </w>:* **then**
      Add all h_{i+1}^1, ..., $h_{i+1}^{|V|}$ expansions of h_i based on cED score to New_Hypotheses
    else
      Add h_i to New_Hypotheses
  Hypotheses =K-best$_{cED}$(New_Hypotheses)
return Hypotheses

FIGURE 4.2: Pseudocode for Synchronized Decoding Algorithm. The higher-level synchronized beam search **SyncBeamSearch** expands segmentation hypotheses by morphemes and prunes them by a combined score of cED, HLLM and LC. The inner character-level modified beam search **ModifiedBeam$_{cED}$** expands hypotheses by characters and prunes them by cED score.

Fusion of cEd with LM Among the approach combining an encoder-decoder framework with a language model, most similar to our approach is the "shallow fusion" of Gulcehre et al. (2017). They integrate a language model into a ED system for a different task of machine translation. There are, however, several important differences in the architectures of our methods.

First of all, the role of the language model is different. Integrating a language model allows Gulcehre et al. (2017) to augment the parallel training data with additional monolingual corpora on the target side. In this way, they add new information about sequencing, not captured in training on parallel data alone. Both components of their system are trained on the same kind of units — characters. As opposed to this, we use a language model to extract more information from the parallel data. We add new information by training the system at two levels: the basic encoder-decoder component is trained on character sequences and the language model component is trained on the sequences of morphemes. In the case of Gulcehre et al. (2017), the use of a language model is motivated by the fact that external monolingual target-side data is almost always universally available. The situation is reversed for the task of morphological segmentation: morphologically segmented corpora are produced manually by experts in the process of linguistic analysis and they tend to be small and expensive. Our approach is motivated by the need to extract as much information as possible from relatively small target-side data sets.

Second, we add a third component to our model which controls for the difference in characters length between the input word string and the output segmentation string. This helps overcome language model preference for a short output.

Last, while Gulcehre et al. (2017) use a language model implemented with recurrent neural networks, we employ a statistical language model, which is better adapted to our settings with small data sets.

Canonical Segmentation As underlined in the introduction, the task of canonical segmentation differs from the task of surface segmentation which has its own history of methods. In the following, we give an overview of the approaches only for the canonical segmentation which is the topic of our study.

Initially, following a long tradition of unsupervised modelling for the surface segmentation, canonical segmentation was approached with unsupervised methods too. Dasgupta and Ng (2007) use heuristic methods to combine unsupervised surface segmentation algorithm of Keshava and Pitler (2006) with the induction of orthographic rules. Naradowsky and Goldwater (2009) use a Bayesian framework to include learning of spelling rules into surface segmentation method of Goldwater, Griffiths, and Johnson (2006). Bergmanis and Goldwater (2017) make a step towards canonical segmentation by extending a log-linear surface segmentation model of Narasimhan, Barzilay, and Jaakkola (2015) where the latter includes semantic information in the unsupervised learning.

More recently, the task of canonical segmentation was casted as a sequence transduction problem and tackled with supervised methods: conditional random fields (Cotterell et al., 2015; Cotterell, Vieira, and Schütze, 2016; Cotterell and Schütze, 2018) and neural ED model (Kann, Cotterell, and Schütze, 2016). As in unsupervised setting, the former approaches build on the previous method for surface segmentation and both build on the supervised CRF-MORPH system of Ruokolainen et al. (2013). The CRF-MORPH system tags each character of a morphologically complex word with one of the tags 'B' for the beginning, 'M' middle, and 'E' end of a segment, and 'S' for a single character segment. The CHIPMUNK model of Cotterell et al. (2015) based on a semi-Markov model extends the CRF-MORPH approach by adding features from stand-alone dictionaries and affix lists. Cotterell, Vieira, and Schütze (2016) tackle canonical segmentation by combining the semi-Markov segmentation model of Cotterell et al. (2015) with a finite-state transduction model for modeling orthographic changes. Kann, Cotterell, and Schütze (2016) improved the results by Cotterell, Vieira, and Schütze (2016) on canonical segmentation by employing the encoder-decoder framework. In their system, they apply a re-ranker to the output of the encoder-decoder system. The re-ranking component is a multilayer perception run on the morphemes embeddings. The morphemes embedding used for this re-ranking model are calculated using additional information from the Aspell dictionaries. Later, Cotterell and Schütze (2018) build on the joint system of Cotterell, Vieira, and Schütze (2016) and achieve state-of-the art results on canonical segmentation. The use extra Wikipedia data to learn word embeddings which they include as additional features into the joint system.

In sum, all previous supervised systems applied to the task of canonical segmentation make use of additional heterogeneous target data. In contrast, our synchronized decoding approach exploits the signal from higher level units in the target side of the parallel train data. In such form, our approach receives higher performance than the previous methods of Cotterell, Vieira, and Schütze (2016) and Kann, Cotterell, and Schütze (2016) relying on additional target data (Section 4.4). In the setting, where we follow previous work and incorporate higher level units signal from additional lexical resources (Section 4.5), our approach leads to the improvement over the previous state-of-the-art canonical segmenter of Cotterell and Schütze (2018).

4.4 Experiment 1: HLLM over Target Side of Parallel Data

In our first experiment, we assess the performance impact of the syncronized decoding on the task of canonical morphological segmentation. To this end, we run our experiments on the canonical segmentation datasets for English, German and Indonesian

released by Cotterell, Vieira, and Schütze (2016)[1]. This dataset allows us to compare our system with the previous state-of-the-art. The corpus for each language is constructed on the 10,000 forms selected at random from a uniform distribution over types. This data is further used to sample 10 splits into 8000 train, 1000 development and 1000 test pairs. We report the numbers on 5 splits to compare to the results of the existing models.

Apart from comparison of our methods to the previous solutions on each language, we perform an additional analysis to assess generalization power of our model. To carry out this task, we report performance on two groups of words unseen in the training. The division into the groups is based on the criteria whether the segmentation of a word contains morphemes unseen in the training. In this way, we directly analyze the effect of capturing regularities in morpheme sequences by HLLM on the overall performance.

The experiments in this section are published in Ruzsics and Samardžić (2017). For comparison, we only report previous state-of-the-art systems which were available at the time of publishing Ruzsics and Samardžić (2017). The performance of the most recent competitive systems are reported in Section 4.5 where we perform experiments with extra data. This provides a more profound comparison to the current systems since they rely on extra data too.

4.4.1 Experimental Setup

Baseline and Comparison As a baseline, we use the basic component of our model (cED), a character-level attention encoder-decoder model with the hyperparameters described below. The architecture of cED is identical to the standard setup described in Section 3.2 except the scoring function for calculating attention component. Instead of the general scoring function (Eq. 3.23), we use MLP scoring function originally proposed by Bahdanau, Cho, and Bengio (2015):

$$\begin{aligned} \alpha_t^i &= \mathbf{v_t}^\intercal \tanh(\mathbf{W}_s \mathbf{s_t} + \mathbf{W}_h \mathbf{h}_i^x) \\ \mathbf{a}_t &= \text{softmax}(\boldsymbol{\alpha}_t) \end{aligned} \tag{4.5}$$

where MLP is a multi-layer perceptron (Eq. 3.8) and α_t^i is alignment score which "compares" current decoder state $\mathbf{s_t}$ at time step t to the encoded input at position i.

The resulting architecture of cED model used in this experiment differs form the one reported in Ruzsics and Samardžić (2017) where we used the ED architecture very close to the one originally proposed by Bahdanau, Cho, and Bengio (2015). The main difference between the current model used in this chapter and the one reported in the paper is a computation path. Concretely, the current setup follows Luong, Pham, and Manning (2015) and has the following computation path: $y_{t-1} \rightarrow \mathbf{s}_t \rightarrow \mathbf{a}_t \rightarrow \mathbf{c}_t \xrightarrow{\mathbf{s}_t} \tilde{\mathbf{s}}_\mathbf{t}$, i.e. the model first updates the decoder state using the previous prediction y_{t-1} (the

[1] ryancotterell.github.io/canonical-segmentation

third line in Eq. 3.21), then uses current decoder state $\mathbf{s}_t$ to calculate attention weights $\mathbf{a}_t$ and context vector $\mathbf{s}_t$ (Eq. 4.5 and Eq. 3.23), then vectors $\mathbf{s}_t$ and $\mathbf{c}_t$ are used to obtain attentional decoder state $\tilde{s}_t$. The last vector is used to make a prediction y_t as detailed in Eq. 3.21. On the other hand, at any time t, Bahdanau, Cho, and Bengio (2015) build from the previous hidden state $\mathbf{s}_{t-1} \rightarrow \mathbf{a}_t \rightarrow \mathbf{c}_t \rightarrow \mathbf{s}_t$, where to update the decoder state, the model first uses previous decoder state $\mathbf{s}_{t-1}$ to calculate attention weights $\mathbf{a}_t$ and context vector $\mathbf{c}_t$ (Eq. 4.5 with $\mathbf{s}_t$ replaced by $\mathbf{s}_{t-1}$, and Eq. 3.23), then uses $\mathbf{c}_t$ and previous prediction y_{t-1} to update the decoder state (the third line in Eq. 3.21, where a concatenation of two vectors $[\mathbf{c}_t, \mathbf{y}_{t-1}]$ is used as an input to RNN). After that, the updated decoder vector $\mathbf{s}_t$ goes through a deep-output and a maxout layer (see exact formulas in Bahdanau, Cho, and Bengio (2015)) before making a prediction y_t.

We also compare the encoder-decoder model to the character-level statistical machine translation (cSMT). This approach is a natural choice in machine translation with small training sets, but no results have been reported so far for the task of canonical segmentation. We used the Moses toolkit with the following settings: distortion is disallowed and build-in MERT optimization is used to optimize the translation model and language model.

As a reference, we compare our results to the joint transduction and segmentation model of Cotterell, Vieira, and Schütze (2016) and the state-of-the-art neural reranker model of Kann, Cotterell, and Schütze (2016). Note, however, that the results cannot be directly compared to these two systems since both models use extra training material in the form of external dictionaries.

Evaluation The evaluation is performed at the level of word tokens using accuracy of the full segmentation. To analyze how well our approach generalizes to process new words, unseen in the training data, we evaluate the performance on subsets of test words. Specifically, in order to analyze the performance of the models on segmenting new words, we distinguish between two subcategories of test words that are not seen in the training corpus: a) new combinations of morphemes already seen during training (*New combinations* category); and b) words that contain unseen morphemes (*New morphemes* category).

Hyperparameters and Implementation The character embeddings of cED system are shared between input (source) and output (target) vocabulary and are set to 100. All LSTM networks have 200 hidden units. We apply an ensemble of 5 cED models, where each model is trained using SGD optimization. The models are trained for a maximum of 30 epochs, possibly stopping earlier if the performance measured on the development set is not improving after 10 epochs. The training examples are shuffled before each epoch. For decoding we use beam width of size 3. We combine 5 cED models by summing up their log probabilities at each character prediction step

(no majority voting). In case of the synchronized decoding, such accumulated sum is combined with the weighted log probability of the HLLM at segment boundaries.

We train the language model HLLM over morpheme sequences using SRILM toolkit.[2] The model is trained on the target side of the parallel corpus, i.e. the canonical segmentations. Following initial experiments, we use morpheme 3-gram language model and apply Kneser-Ney smoothing (Section 3.1.1).

To set the hyperparameters corresponding to weights of language model and length control in Eq. 4.4, we use the Z-MERT tool[3] which we integrate into our implementation. As the objective for the MERT weight optimization we use accuracy on the development set. In order to generate n-best list for the optimization procedure, we use beam width of size 3.

Our implementation of the cED model in DyNet framework is shared in an online repository.[4] In this repository we also share the code for synchronized decoding to combine neural cED model and statistical language model HLLM.

4.4.2 Results and Discussion

The performance of our multi-level model cED+HLLM together with the two baseline models is reported for English, German, Indonesian in Table 4.1. We show the average results over five splits for each language along with the standard deviation (in brackets).

					+ASPELL	
	No of, %	cED	cED+HLLM	SMT	RR*	Joint*
English		80. (1.)	80. (1.)	73. (2.)	**81.** (1.)	73. (2.)
New morph.	70.1 (0.6)	**80.** (1.)	79. (1.)			
New comb.	28.6 (0.7)	81. (2.)	**87.** (2.)			
German		80. (1.)	**82.** (1.)	76. (2.)	80. (1.)	59. (3.)
New morph.	75.1 (0.5)	**80.** (0.)	79. (1.)			
New comb.	24.3 (0.6)	80. (2.)	**90.** (2.)			
Indonesian		94. (1.)	**97.** (1.)	94. (1.)	95. (1.)	90. (1.)
New morph.	20.6 (0.7)	**91.** (4.)	88. (3.)			
New comb.	79.4 (0.7)	95. (0.)	**99.** (0.)			

TABLE 4.1: Performance on the task of canonical segmentation (Word accuracy and standard deviation averaged over 5 splits, the rounding schemes of previously published results are applied.). RR* - neural reranker model of Kann, Cotterell, and Schütze (2016). Joint* - joint transduction and segmentation model of Cotterell, Vieira, and Schütze (2016). The subcategories of the data are described in Section 4.4.1.

We observe that out of the two baseline models, cED and cSMT, cED performs on average better although their accuracy is the same in the case of Indonesian.

For reference, we also show the results of the joint model of Cotterell, Vieira, and Schütze (2016) and the state-of-the-art neural reranker model of Kann, Cotterell, and Schütze (2016) which are available for these data sets. We can see that our approach

[2] http://www.speech.sri.com/projects/srilm/
[3] http://www.cs.jhu.edu/~ozaidan/zmert/
[4] https://github.com/tatyana-ruzsics/uzh-corpuslab-gen-norm

(cED+HLLM) gives an improvement of 2% for both German and Indonesian over the state-of-the-art performance while we do not employ extra information from external dictionaries. Note that the languages for which we improve the state-of-the-art are morphologically richer than English.

Our fused approach gives an improvement from 2% to 3% over the stronger cED baseline. The bigger improvement for Indonesian could be attributed to more regular concatenative patterns. This observation becomes even more evident in our next analysis which assesses how well our model generalizes to unseen words.

Generalization Analysis Table 4.1 shows the results on the categories of the words not seen in the training for each languages. Since the datasets are constructed based on word types rather than on word tokens, the category of unseen words makes up for 99% of the test sets in this corpora. We observe that the HLLM component helps to correct errors for the words consisting of new combinations of seen morphemes at the expense of a slight drop of the performance on the words consisting of new morphemes. Among the three languages, accuracy of cED+HLLM model on the *New combinations* category is almost 100% for Indonesian, while it is only close to 90% for English and German. Analyzing the errors of the models on this category, we observe that in most cases wrong segmentations in English and German are due to orthographic changes on the segmentation boundaries. For example, the model segments English word *dissension* as *dissend*|*ion* while the true segmentation is *dissent*|*ion*. Another example is a segmentation *numeric*|*al* instead of *numberic*|*al* for the word *numerical*. Such cases could be attributed to less regular orthographic changes which the model fails to recognize already at the character generation stage by cED component. This results in no correct partial hypothesis which could have been given a higher weight by HLLM. Less regular orthographic changes in English and German seem to drag performance down on the *New morphemes* category too, compared with the performance of this category on Indonesian.

Between the two categories of unseen data, the performance on the *New morphemes* category is lower than that of the *New combinations* category, in all three languages. In case of Indonesian, around 80% of new words (an average over five splits) belongs to the latter category, while its share is only around 25% for German and English. The overall lower performance for English and German thus might be due to the less regular patterns and more unseen roots in the training data. In the next section, we investigate to which degree we can improve performance for the new words consisting of unseen morphemes by integrating additional language data.

4.5 Experiment 2: HLLM over Extra Out-of-Domain Data

In the previous experiment, we found that our synchronized decoding method is particularly helpful in segmenting unseen words consisting of new combinations of

seen morphemes. In this section, we investigate whether integration of more noisy language data can improve performance of our model on segmenting unseen words consisting of new morphemes. To this end, we propose to employ additional language data for training of HLLM component. Extra target data in the form of manually segmented words is an expensive resource to produce. Therefore, we apply a more realistic setting where we use more noisy target data in the form of dictionaries. We expect such approach to target directly the category of new words which consist of unseen morphemes, especially unseen roots.

We test our approach on canonical segmentation datasets for English and German used in Section 4.4 where performance on the category of unseen words consisting of new morphemes especially suffers from data sparsity. Both sets consist of 10,000 word types split randomly 10 times into 8,000 train, 1,000 development and 1,000 test pairs. We report the numbers on 5 splits and 10 splits to compare to the results of the existing models. As an additional target data we use Aspell dictionaries[5], which amount to 120K word types for English and 365K word types for German.

4.5.1 Experimental Setup

We apply our multi-level method (cED+HLLM) where we train a target-side HLLM over morpheme sequences in the original train target data extended with additional list of words from Aspell. To analyze how our method performs on unseen word types, we also add the results for plain cED model and cED+HLLM model where HLLM is trained only over morpheme sequences in the original train target data. For direct comparison, we show the results of the neural reranker model of Kann, Cotterell, and Schütze (2016), joint transduction and segmentation model of Cotterell, Vieira, and Schütze (2016). Both systems employ ASPELL data too. We also show the results of the joint transduction and segmentation model with word embeddings (trained on Wikipedia data) of Cotterell and Schütze (2018) which is not directly comparable to our and previous methods due to use of more additional data.

The configuration of the basic component of our model (cED), hyperparameters for training cED and HLLM as well as evaluation approach stay the same as in the previous experiment with extra target data (Section 4.4.1).

4.5.2 Results and Discussion

The results of the segmentation experiments are presented in Table 4.2. We observe that our multi-level model cED+HLLM with additional Aspell target data reaches 83% accuracy for both English and German improving over the other models which rely on the additional data, including the state-of-the-art Joint+Vec model which uses even extra data from Wikipedia.

[5] http://cistern.cis.lmu.de/chipmunk/supplement.tar.gz

				+ASPELL			
	No of, %	cED	cED +HLLM	RR*	Joint*	cED +HLLM	+WIKI Joint+Vec*
English-5		80. (1.)	80. (1.)	81. (1.)		**82.** (1.)	
English-10:		79. (1.)	80. (1.)		77. (1.3)	**83.** (1.)	82. (2.)
New morph.	69.3 (1.2)	80. (1.)	79. (1.)			**83.** (1.)	
New comb.	29.3 (1.1)	80. (3.)	**86.** (3.)			84. (3.)	
German-5		80. (0.)	82. (1.)	80. (1.)		**83.** (1.)	
German-10:		80. (1.)	82. (1.)		79. (1.)	**83.** (1.)	82. (1.)
New morph.	75.3 (0.7)	80. (1.)	79. (1.)			**83.** (1.)	
New comb.	24.2 (0.7)	79. (3.)	**89.** (2.)			83. (3.)	

TABLE 4.2: Performance on the task of canonical segmentation for English and German in the setting with extra data (Word accuracy and standard deviation averaged over 5 splits (English-5, German-5) and over 10 splits (English-10, German-10), the rounding schemes of previously published results are applied.). RR* - neural reranker model of Kann, Cotterell, and Schütze (2016). Joint* - joint transduction and segmentation model of Cotterell, Vieira, and Schütze (2016) (results avereged over 10 splits are reported in Cotterell and Schütze (2018)). Joint + Vec* - joint transduction and segmentation model with word embeddings (trained on Wikipedia data) of Cotterell and Schütze (2018). The subcategories of the data are described in Section 4.4.1. The performance on subcategories is averaged over 10 splits.

Generalization Analysis We analyze the impact of additional data in our solution by considering how well our model generalizes to unseen words in the two categories of *New morphemes* and *New morphemes.*

Although cED+HLLM model with HLLM trained *only* on the target side of the parallel train data shows improvement over the plain cED system in the *New combinations* category, the use of additional target data helps with the performance in the *New morphemes* category mainly due to unseen roots that can be found in additional data. We observe that with employment of additional data results in a slight drop of the performance on the *New combinations* category. However, cED+HLLM model with HLLM trained over extra data achieves overall the highest result for both languages. This improvement can be explained by a high weight of the *New morphemes* category in the data structure where additional data is the most helpful.

4.6 Summary

Learning internal word structure has recently been recognized as an important step in various multilingual processing tasks and in theoretical language comparison. In this chapter, we present a multi-level neural encoder-decoder model for learning canonical morphological segmentation. Our model combines character-level sequence-to-sequence transformation with a language model over canonical segments. We fuse the two components during the decoding phase with our novel synchronized decoding approach. Our method allows learning from the same data but combining signal on two levels, characters and segments. In our experiment on three languages, we obtain up to 3% improvement over a strong character-level encoder-decoder baseline. Our

model outperforms the previous state-of-the-art for three languages while eliminating the need for external resources such as large dictionaries for languages with regular concatenative patterns as in Indonesian.

Besides assessing our method's generalization ability across different languages, we perform additional analysis to test how well it generalizes to new data. To this end, we introduce two different categories of unseen test words. The basis for division into two classes is whether segmentation of a word contains morphemes unseen in the training data. We find that our approach leads to improvement of performance on unseen input words for which segmentation consists of a new combination of segments seen in the training data. To achieve further improvements in the other category, we propose integrating more noisy language data in our approach. We show that such approach leads to further improvements on unseen input words whose segmentation contain unseen segments, especially new roots. By integrating only dictionary data, our model achieves state-of-the-art results on English and German, while using less extra language data than the previous best solutions.

Benefits of our synchronized decoding method expand beyond the explicit task of words segmentation. The method can be portable to any other upstream task where segmentation is more implicit. In the next chapter, we demonstrate such portability for the task of writing normalization.

Chapter 5

Hierarchical Encoding with Synchronized Decoding for Text Normalization

5.1 Introduction

In this study, we consider the form of upstream processing which aims to remove writing variation in text. The high degree of variation of writing the same word can be attributed to several factors. Firstly, the lack of a standardized spelling which can be further complicated by dialectological differences in the same language. For example, in Swiss German dialects, such factors lead to the numerous local variants of the same word. To illustrate, a word *viel* ('much') can appear as *viel*, *viil*, *vill*, *viu* and many other potential variations. Secondly, noise in a written text can appear when it is used in computer-mediated communication (CMC). CMC is characterized by various peculiarities, such as vowel reduplication and unconventional abbreviations, which increase variation. Finally, variation can be due to a historical factor: spellings change over time, but also vary within a single time period and even within a single author, since orthography only became standardized in many languages fairly recently. For example, the modern English word *said* might be realized in historical texts as *sayed*, *seyd*, *said*, *sayd*, etc. Normalizing text in dialectological and historical documents makes them more searchable, which is valuable for theoretical research. On the other hand, it helps to improve the performance of downstream NLP tools by reducing data sparseness.

We focus on the task of normalizing Swiss German text, illustrated in the following example:

(17) Writing Normalization Task (Swiss German)

Source form (Swiss German)		Normalized form (Standard German)
viil 'much'	▷	*viel*
viu 'much'	▷	*viel*
vill 'much'	▷	*viel*

The goal of Swiss German normalization is to reduce the variation by mapping words written in Swiss German to their canonical form in Standard German. The factors contributing to the variation in Swiss German are described in more detail in Section 2.2.2. The task of Swiss German normalization can be cast as character sequence transformation and therefore can be solved with a character-level encoder-decoder framework (cED), introduced in Section 3.2. Our goal in this study is to enhance the performance of a basic cED system on this task. To this end, we propose two modifications to the cED framework which address different challenges of normalization: generation of fluent canonical text versus ambiguity resolution.

Generation of fluent canonical text Similarly to the task of canonical segmentation considered in the previous study, the goal of the normalization is to tackle data sparseness by reducing variation in the text. Writing normalization maps orthographic words to their canonical form, while canonical segmentation reduces such canonical word form to a sequence of canonical morphemes. There is another notable similarity which inspires us to adapt our multi-level approach developed for canonical segmentation in Chapter 4 in the current study.

Specifically to Swiss German normalization, although most of the mapped sequences are pairs of single words (one-to-one mappings), there are also many contracted forms corresponding to multiple normalized words (one-to-many mappings). For example, Swiss German word *hämmers* 'we have it' is normalized as a sequence of words *haben wir es* in Standard German. To exploit regularities from canonical word sequences, we propose integrating a word-level language model into the cED system at the decoding stage. The language model is trained over word sequences on the target side. To combine the scores of such higher-level language model (HLLM) with the one produced by the basic cED model, we adapt our synchronization mechanism introduced (Chapter 4), resulting in a multi-level decoding.

Besides targeting one-to-many mapping, we expect our synchronized approach to improve on one-to-one transformations too. As we showed in the previous study, further improvements in the processing of unseen words consisting of segments not seen in the training can be achieved by the integration of extra data. Such data is used to extend the target side of parallel data for training HLLM component. In the case of the writing normalization task, this approach can exploit additional texts written in the canonical language. These resources are often widely available due to the status of canonical language as an orthographic standard to be used in writing. We expect that the integration of extra texts in canonical language will result in generating more fluent output, i.e. normalized words, or sequences of words, which are highly probable in the canonical language. In this way, the resulting multi-level model is expected to refine normalizations generated by cED system. Specifically, the basic cED component learns local character transformations and therefore, it might produce an output that is not a proper word in a canonical language, despite being a likely sequence of character.

Ambiguity resolution Another problem which arises in text normalization is ambiguity: in some cases, one word in Swiss German can be normalized in two or more different ways. For example, the word *lüüt* can be normalized either as *läuten* 'to ring' or Leute 'people', depending on its context. Ambiguity in the text normalization is an instantiation of word sense ambiguity challenge which is ubiquitous in computational linguistics.[1]

Word sense ambiguity occurs when the same word can be used to express different meanings. For example, two senses of the English word *book* become clear when used in context:

(18) Two senses of English word *book*

book[1]	... *book* that flight ...
book[2]	... hand me that *book* ...

In practical applications, e.g. translating from English to another language, there is a need to disambiguate the different uses of a word. The insight of introducing context information that underlies modern approaches for addressing ambiguity was first articulated by Weaver (1955) for the purpose of machine translation: "If one examines the words in a book, one at a time as through an opaque mask with a hole in it one word wide, then it is obviously impossible to determine, one at a time, the meaning of the words. ...But if one lengthens the slit in the opaque mask, until one can see not only the central word in question but also say N words on either side, then if N is large enough one can unambiguously decide the meaning of the central word. ...The practical question is: What minimum value of N will, at least in a tolerable fraction of cases, lead to the correct choice of meaning for the central word."

One common way in NLP to resolve ambiguity is to extract information from the context, for example, in the form of Part-of-speech (PoS) tags. PoS tagging is the process of assigning a PoS label to each word in a sequence of words. For example, PoS tag for the word *book* in its first sense in Ex. 18 is *Verb*, while the word in the second sense has the tag *Noun*. After PoS tagging preprocessing step, assigned tags can be further integrated into a downstream system to address the ambiguity.

Disambiguation for word senses can be challenging and not always solvable with PoS tags. For example, German uses two distinct words for what in English would be called a wall: *Wand* for walls inside a building, and *Mauer* for walls outside a building. Where English uses the word *brother* for any male sibling, Chinese and many other languages have distinct words for older brother and younger brother: for example, Mandarin uses *gege* and *didi*, respectively (Jurafsky and Martin, 2020). In all these cases, translating words like *wall* or *brother* from English would require a

[1] Ambiguity problem arises in the task of morphological segmentation too. For example, Turkish word *adamı* can be segmented as *adam*|*ı* (an accusative form of 'men') or *ada*|*m*|*ı* (an accusative form of 'my island'). We did not attempt to target ambiguity when developing our methods in Chapter 4 due to the type-based nature of the canonical segmentation datasets: only one correct segmentation is selected as gold for each word form. However, we argue that our solutions proposed to resolve ambiguity in the current study can be adapted to develop segmentation methods for datasets where segmentation ambiguity is present.

different kind of specialization than PoS tags in order to disambiguate the different uses of a word.

To overcome the problem of ambiguity in text normalization, we propose and test several modifications to the basic cED system. Each modification extracts and integrates context signal in a different form. Apart from integrating context information as PoS tags, we explore methods to integrate a context signal in the form hierarchical RNN language model on the encoder side. The latter modification leads to a multi-level hierarchical encoder in cED system. It is applied to a sentence in which an input word is used. Besides providing character-level representation of the input word through the character-level encoder component of the cED, it allows extracting context information from the sentence on two levels: characters versus words.

Analysis of model behaviour In our approach, we combine two multi-level modifications to the cED system described above. Adaptation of our synchronized decoding is expected to improve the fluency of the generated canonical text. Our modification for the integration of the context signal targets ambiguity in normalization. Crucially, a combination of the two approaches is expected to work in a complementary manner: the resulting multi-level system is designed to jointly improve the fluency of the output and resolve cases of ambiguity. To test these hypotheses, we perform a detailed quantitative analysis in our experiments. To this end, we follow and extend the analysis approach proposed in Chapter 4 and introduce specific categories of test words. Such setup allows us to perform additional qualitative analysis of model behaviour. In particular, we investigate to which extent our proposed modifications to cED model are sufficient to improve fluency and resolve the ambiguity.

Outline of the study In the remaining of this chapter, we first present our modifications to a base cED model for addressing ambiguity resolution and output fluency in text normalization. Then, we compare our approach to the previous work in Section 5.3. We proceed by describing the first experiment (Section 5.4) designed to a) test overall performance of our system and compare it to the previous solution and competitive cED baseline; b) quantitatively and qualitatively analyze complementarity of our two modifications; c) test whether a context signal in the form of PoS tags is sufficient to successfully resolve the cases of ambiguity. Finally, we proceed to the second experiment (Section 5.5), where we test a) whether summarizing context information with higher-level RNN networks is more efficient in resolving ambiguity than explicit PoS tags; b) whether a combination of two types of context signal works complementary for this purpose.

5.2 Hierarchical Encoder-Decoder for Text Normalization

In the following sections, we describe the details of our multi-level modifications for a standard character-level encoder-decoder framework cED, introduced in Section 3.2, to the task of text normalization.

The normalization task can be cast as a transformation of the input sequence of characters to the output sequence of characters. More formally, we define two alphabet sets, Σ consisting of the character symbols that form the source sequences (second column in Table 5.1) and Σ_n of the character symbols that form the normalized sequences (third column in Table 5.1). Then, our task is to learn a mapping from an original character sequence $x \in \Sigma^*$ to its normalized form $y \in \Sigma_n^*$. For example, in the task of Swiss German normalization, the input word *viil* (as well as its variants, e.g. *vill*) has to be mapped to its normalized form *viel* 'much'. Specifically to Swiss German normalization, most of the mapped sequences are pairs of single words (one-to-one alignments) as shown in the first section of Table 5.1. There are also many contracted forms corresponding to multiple normalized words (one-to-many alignments). These are typically verb forms or prepositions merged with subject and object clitics, as illustrated in the second section of Table 5.1.

Following notation introduced in Section 3.2, we denote by V_x and V_y input and output vocabulary, correspondingly, where both vocabulary sets include output symbols from the alphabets Σ and, respectively, Σ_n (possibly limited to a number of most frequent items), OOV symbol and special symbols for beginning (<s>) and end-of-sequence (</s>) padding.

In our approach, we combine two methods to adapt a basic cED model to the normalization task. The basic cED system takes as input the source form, e.g. *viil* 'much', and learns a mapping to its normalized form *viel*. Our first method modifies the decoding stage of the plain cED system, that has been already pretrained for the task. Specifically, we adapt the synchronized decoding mechanism, introduced in Chapter 4 and integrate an additional language model (HLLM) pretrained over higher-level units over the target side of the data. For the purpose of the task of normalization, we consider separate words as higher-level units as opposed to lower-level processing units - characters - of cED component. Such approach allows us to incorporate more target side data and add more fluency to the cED system output. This is achieved by guiding the cED generation process during decoding through synchronizing cED and HLLM scores at word boundaries.

The synchronized decoding process specifically targets the cases of one-to-many alignments. In addition, it results in rescoring one-word hypotheses of the cED system, which occur in one-to-one alignment units. The details of the decoding approach are described in Section 5.2.1.

In our second method we consider the integration of context signal to the neural system in order to deal with ambiguity problem arising in normalization: in some

TABLE 5.1: Examples of aligned token sequences in the corpus of normalized Swiss German WhatsApp messages (Stark, Ueberwasser, and Göhring, 2014; Ueberwasser and Stark, 2017).

alignment type	source form	normalized form	English gloss	PoS
one-to-one	viil	viel	much	PIAT
	vill	viel	much	PIAT
	lüüt	Leute	people	NN
	lüüt	läuten	to ring	VVFIN
	vor	vor	before; in front of	APPR
one-to-many	vor	von der	from the; of the	APPR+ART
	hämmers	haben wir es	have we it	VVFIN+PPER+PPER

cases, one word can be normalized in several different ways. One way to summarize the sequential source context is to use a (manually annotated) PoS tag of the input word. Another possibility is to include the surrounding words as features and extract context signal using a hierarchical bi-RNN, originally proposed by Ling et al. (2015a). In the latter approach, a lower-level bi-RNN encodes the surrounding words on a character level and a higher-level bi-RNN summarizes the entire context by reading the resulting sequences of the character embeddings to the left and to the right of the input word (bi-RNN architecture is introduced in the context of the encoder-decoder framework in Section 3.2). In this way, both character-level and word-level signals from the source sequential context are included.

We make use of both possibilities for extracting context signal. We first consider two separate paths, cED+PoS and cED+HLRNN, where the two kinds of the context information are given separately as input to the cED decoder. Next, we combine the two types of the context signal in a flat manner (cED+HLRNN+PoS) and in a hierarchical fashion (cED+HLRNN+Predicted PoS). The latter model first uses context to predict a PoS tag, then uses the context together with the predicted PoS tag for the task prediction. The former model can be considered as a ceiling for the hierarchical model cED+HLRNN+Predicted PoS. The details of the architectures integrating the source context are described in Section 5.2.2.

Each of the proposed enhancements to the plain cED system targets specific phenomena in the corpus. Fluency and, especially, one-to-many alignments are addressed by the synchronized decoding. On the other hand, the additional context signal features address the problem of processing ambiguous words. Moreover, the combination of the two approaches is expected to work complementary in the cases which exhibit both phenomena. For example, the input word *vor* can be either normalized as the preposition *vor* ('before'; 'in front of') or as a preposition followed by an article, as in *von der* 'from the'.

5.2.1 Adapting Synchronized Decoding: cED+HLLM

Our first approach designed to achieve fluency in writing normalisation is an adaptation of the synchronized decoding mechanism for integrating a HLLM into the cED

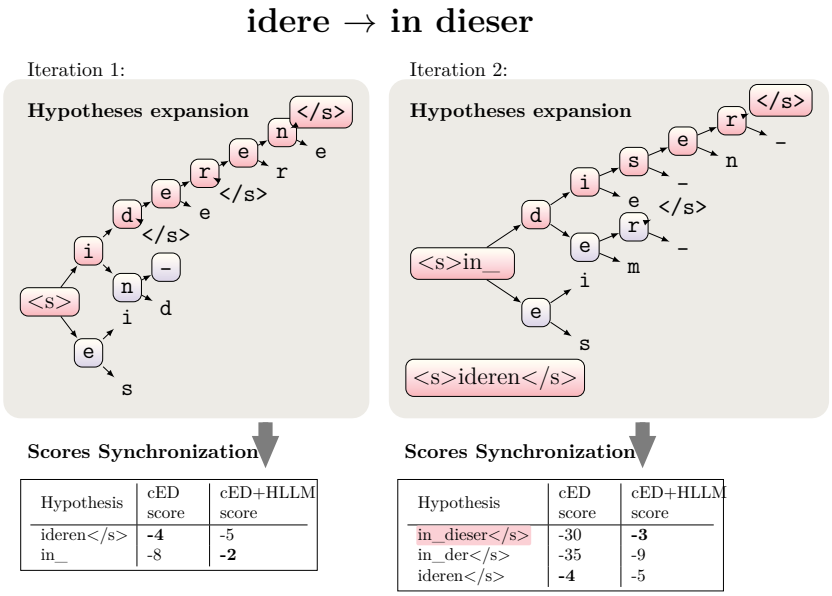

FIGURE 5.1: Synchronized decoding algorithm for searching normalization of the Swiss German word *idere* 'in this'. Both word-level and character-level beam search are run with beam width 2. The search involves two iterations of the higher-level beam search. Selected expansions at each character-level iteration of the lower-level beam search are marked with boxes. Out of two selected expansions, a character with a higher cED score is coloured in red, while the other one is coloured in blue. The scores are negative log-probabilities.

system, proposed for the task of canonical segmentation in Chapter 4. Before the integration, we assume that a cED model and a HLLM are trained separately. The cED model is trained on character sequences in a parallel corpus consisting of aligned source words and their normalized forms (as shown in Table 5.1). We augment this model with an additional HLLM, separately trained over the word sequences in the target side of the corpus. Therefore, the additional HLLM brings the frequency signal from higher-level units of the target data (words), while the cED system operates on characters. The scores of both components are fused, at the decoding stage through the synchronized decoding algorithm Section 4.2. We adapt the algorithm to the task of writing normalization by fusing the scores on the words boundaries, instead of segment boundaries in the canonical segmentation task. The synchronized decoding is performed with a combination of two beam searches run on a different level. The word-level beam search works normally by expanding hypothesis by whole words with a combination of cED and HLLM scores. This is achieved with the help of a modified character-level beam search which gets a partial hypothesis from the higher-level search and expands it by characters using cED score. The difference of the character-level beam search from a standard beam search is the termination condition: the search stops at the synchronization point, i.e. where all beam width-number of hypothesis expansions end with the end-of-sequence or word boundary symbol, where the latter is a space character: ' ' $\in V_y$. Then, the expansions are passed back to the word-level beam search for rescoring.

Figure 5.1 illustrates the process of synchronized decoding for the source word *idere*, whose correct normalization form is *in dieser* 'in this'. Both word-level and character-level beam search are run with beam width 2. In the first iteration of the high-level beam search, beginning-of-sequence symbol is passed to a lower-level beam-search for expansion. The resulting expansion *ideren</s>* gets a greater probability by the cED model, than the second expansion *in_*. In a pure cED model, this first iteration would already lead to search termination and wrong prediction (*ideren</s>*). However, when the expansions are passed back to the higher-level search and the word-level HLLM is used for rescoring, *in_* gets a greater probability than *ideren</s>*, since the HLLM is aware that the latter is not a word, according to the train set. In the second iteration, since the hypothesis *ideren</s>* already ends with the end-of-word symbol, only the second one, *in_*, is passed to the lower-beam search for further expansion. This results in two expansions which are passed back to the higher-level search. After rescoring three expansions on this step, the search process ends and the sequence *in_ dieser</s>* is correctly given the greatest probability by the cED+HLLM model.

5.2.2 Four Variants of Context-sensitive Encoder-Decoder

We consider several architectures for incorporating contextual information into the plain cED system. In particular, each of these methods incorporates contextual features associated with the whole input word x into conditional probability at each character-level prediction step of the plain cED model (Eq. 3.21). For convenience, we repeat the formulation of the conditional probability here:

$$\begin{aligned} P(y_{t+1}|y_{1:t}, x) &= \text{softmax}(\mathbf{W}_o\tilde{\mathbf{s}}_{t+1} + \mathbf{b}_o) \\ \tilde{\mathbf{s}}_{t+1} &= \tanh(\mathbf{W}_a[O(\mathbf{s}_{t+1}); \mathbf{c}_{t+1}]) \\ \mathbf{s}_{t+1} &= R(\mathbf{s}_t, \hat{\mathbf{y}}_t) \\ \hat{y}_t &\sim P(y_t|y_{1:t-1}, x) \end{aligned} \tag{5.1}$$

We integrate context features in each model by modifying the MLP component for calculating attentional decoder state $\tilde{\mathbf{s}}_\mathbf{t}$ (the second line of Eq. 5.1).

cED+PoS In our first system, we assume that the contextual input is represented only by a PoS-tag of the input word. In addition to the two vocabularies that contain the source V_x and target V_y characters, we consider a vocabulary of the possible PoS tags V_f. Our task in this setting is to learn a mapping from an input pair $(x, \quad f_x = f_1 + ... + f_k)$ of a source character sequence $x \in V_x^*$ and its PoS feature f_x (possibly consisting of one or more tags $f_i \in V_f$) to its target normalized form $y \in V_y^*$. It is specific to the task of normalization that a PoS feature may consist of several tags: such cases appear when the mapping from x to y belongs to the category of one-to-many alignments (Table 5.1).

We embed the PoS tags $f_i \in V_f$ into their vector representations $\mathbf{f}_i$, which are learned by the system. In cases where the feature input is a composition, i.e. it consists of several PoS tags $f_1 + ... + f_k$, we use an average of the corresponding vector embeddings $\mathbf{f}_x = (\mathbf{f}_1 + ... + \mathbf{f}_k)/k$ as representation. We then adapt the plain cED system and feed the PoS feature, together with the current decoder hidden state $\mathbf{s}_t$ and the current context vector $\mathbf{c}_t$, into the attentional decoder layer, in order to obtain attentional decoder state at prediction step t as follows:

$$\tilde{\mathbf{s}}_\mathbf{t} = \tanh(\mathbf{W}_a[O(\mathbf{s}_t); \mathbf{c}_t; \mathbf{f}_x]) \tag{5.2}$$

where dimension of the matrix $\mathbf{W}_a$ is expanded accordingly. Since the decoder operates at the character level, each prediction step t corresponds to the output of one character (Eq. 5.1). Therefore, the same PoS feature f_x associated with the entire word x is used for the prediction of each one of its characters y_t. In the following, we will refer to both types of features, individual tags and compositional features, as PoS features or PoS tags interchangeably.

cED+HLRNN In the second system, we assume that the input is a pair of a source word x together with its context $s = (s_1, \ldots, s_{i-1}, x, s_{i+1}, \ldots s_{n_s})$ defined as the sentence where the word $x = s_i$ appear in a text. This model does not incorporate PoS tags and uses only the raw context represented by the sequence of words in s.

We encode the context s for the word $x = s_i$ using a hierarchical bi-RNN architecture. First, a lower-level bi-RNN encodes separately all the words in the sentence s on the character-level into a sequence of lower hidden states. We assign to each word its character representation consisting of the last forward and last backward lower-level RNN states. The sequence of words $s = s_1, \ldots, s_{n_s}$ is then transformed into sequence of vectors $\mathbf{e}^s_1, \ldots, \mathbf{e}^s_{n_s}$. Following notations for bi-directional RNN encoding in Eq. 3.20, embedding of the context word at position j is a concatenation of two vectors:

$$\mathbf{e}^s_j = [\overrightarrow{f}(\mathbf{p}_{1:n_{s_j}})_{n_{s_j}}; \overleftarrow{f}(\mathbf{p}_{n_{s_j}:1})_{n_{s_j}}] \tag{5.3}$$

where $s_j = p_1, \ldots, p_{n_{s_j}}, p_k \in V_x$ is a character sequence of the context word s_j. At this point, each word embedding $\mathbf{e}^s_j$ is only aware of the character sequences within the word itself but not yet informed about neighbouring words.

A higher-level bi-RNN encodes the sequence of the character representations of the context words $\mathbf{e}^s_1, \ldots, \mathbf{e}^s_{n_s}$ into a sequence of higher hidden states $\mathbf{H}^s_1, \ldots, \mathbf{H}^s_{n_s}$. We assign a contextual representation to the word at position j using its higher hidden state, i.e. concatenating the forward and backward higher hidden states at this position:

$$\mathbf{H}^s_j = [\overrightarrow{f}(\mathbf{e}^s_{1:n_s})_j; \overleftarrow{f}(\mathbf{e}^s_{n_s:1})_j] \tag{5.4}$$

We adapt the plain cED system to feed the context representation $H^x = H^s_i$ of the input word x, together with the current decoder hidden state $\mathbf{s}_t$ and the current

context vector $\mathbf{c}_t$, in order to obtain attentional decoder state at prediction step t as follows:

$$\tilde{\mathbf{s}}_\mathbf{t} = \tanh(\mathbf{W}_a[O(\mathbf{s}_t); \mathbf{c}_t; \mathbf{H}^x]) \tag{5.5}$$

which is then used to predict the next output character according to Eq. 5.1.

cED+HLRNN+PoS The next system combines together the PoS-tag input and context signal extracted with hierarchical bi-RNN. This system learns a mapping from an input pair (x, f_x, s) of a source word $x \in V_x^*$, its PoS feature $f_x \in V_f$ and context $s = (s_1, \ldots, s_{i-1}, x, s_{i+1}, \ldots s_{n_s})$ to its target normalized form $y \in V_y^*$. We use the methods from the previous systems to encode the input and modify the layer for calculating attentional decoder state as follows:

$$\tilde{\mathbf{s}}_\mathbf{t} = \mathsf{tanh}(\mathbf{W}_a[O(\mathbf{s}_t); \mathbf{c}_t; \mathbf{f}_x; \mathbf{H}^x]) \tag{5.6}$$

In this system, we assume that the PoS-tag input is provided at test time.

cED+HLRNN+Predicted_PoS The previous version of the context system, cED+HLRNN+PoS, results in an unrealistic scenario since PoS-features are not necessarily available in the data. We propose another version, cED+HLRNN+Predicted_PoS, which targets a more realistic setup. This system is trained to predict both the tag of the input and its normalized form. It thus learns a mapping from an input pair (x, s) of a source word $x \in V_x^*$ and its context s to the pair (y, f_x) of the normalized form $y \in V_y^*$ and PoS-feature $f_x \in \tilde{V}_f$. The PoS-features vocabulary $\tilde{V}_f$ contains all the standalone tags in the tag vocabulary V_f as well as all compositional tags $f_1 + ... + f_k$, $f_i \in V_f$ observed in the train set. The prediction step is sequential: the system first predicts the feature f_x according to the distribution over features in $\tilde{V}_f$

$$p(f_x|x) = \mathsf{softmax}(\mathbf{W}_f\mathbf{H}^x) \tag{5.7}$$

where $\mathbf{W}_f$ is a learned parameter. In the next step, the system uses the embedding of the predicted PoS-feature in predicting the normalized form as in (5.6). At the train time, we use the gold tags for the prediction of the normalized form.

The parameters of the model cED+HLRNN+Predicted_PoS are trained by minimizing a combined cross-entropy loss:

$$\mathcal{L}^{CE}(\Theta) = -\sum_{(x,f_x,s,y)} \left(\alpha \log p(f_x|x,s) + \sum_t \log p(y_t|y_{1:t-1}, x, f_x, s) \right) \tag{5.8}$$

where hyperparameter α is optimized on a development set.

Although the models we propose in this section are designed for text normalization, they allow portability to other tasks. We review other methods for text

normalization as well as related work where similar approaches to context signal extraction and integration into the encoder-decoder framework proved to be effective for solving other upstream processing tasks.

5.3 Related Work

In this section, we first review previous methods proposed for text normalization, focusing on the details of neural approaches. Then, we discuss how previous solutions address inclusion of the target data and context signal to solve this task. Finally, we discuss how our context models, proposed in Section 5.2.2, relate to the previous work on context integration into the encoder-decoder system for other upstream methods.

Writing Normalization Writing normalization problem was initially addressed in historical text normalization with automatic induction of rules (Reffle, 2011; Bollmann et al., 2012) or similarity-based form matching inspired by spellchecking (Baron and Rayson, 2008; Pettersson, Megyesi, and Nivre, 2013). A major breakthrough in performing the task was achieved when it was approached as a case of character sequence transformation and tackled with character-level statistical machine translation (SMT) in computer-mediated communication (De Clercq et al., 2013; Ljubesic, Erjavec, and Fiser, 2014), historical texts (Pettersson, Megyesi, and Nivre, 2014) and dialects (Samardžić, Scherrer, and Glaser, 2015; Scherrer and Ljubešić, 2016).

After the introduction of neural encoder-decoder methods, there have been numerous efforts to apply this paradigm in a character-lever setting to text normalization. Bollmann and Søgaard (2016) apply RNN encoder-decoder framework to normalization of historical German texts written at different time periods. Bollmann, Bingel, and Søgaard (2017) improve performance of such system further by employing multi-task learning technique: apart from learning to normalize, this system is trained to learn grapheme-to-phoneme mapping (word pronunciation) for German words as an auxiliary task. Using a different dataset for historical German normalization, Korchagina (2017) adapts encoder-decoder system with convolution networks of (Lee, Cho, and Hofmann, 2017) to the task. Tang et al. (2018) provide an extensive evaluation over a range of encoder-decoder systems applied to historical normalization for 5 languages. The systems differ in attention mechanisms (encoder-decoder attention versus self-attention), processing units (character versus subwords) and other architectural aspects. For the same datasets, Robertson and Goldwater (2018) test intrinsic and extrinsic approaches to evaluation of encoder-decoder models. For normalizing dialect data, results for encoder-decoder system are reported in Honnet et al. (2018). In our approach, we follow this general line of work by integrating two components into the encoder-decoder system: our synchronized decoding approach (introduced in Chapter 4) for reaching a more fluent output and context signal information to address word level ambiguity.

In SMT, integration of additional data on the target side is a standard technique to improve fluency of the output. This method was successfully used for writing normalization by Scherrer and Ljubešić (2016) and Honnet et al. (2018): the former report results on Archimob corpus (Samardzic, Scherrer, and Glaser, 2016) which combines several dialects of Swiss German, while the latter apply this approach to several datasets, each representing a separate dialect in Swiss German. We underline the major difference between these approach and our synchronized decoding: our decoding approach is multi-level, combining scores of a character-level translation component cED and word-level language model HLLM on word boundaries. In contrast, in SMT system, where additional target data is used to train a language model component, decoding is performed only at the character-level, i.e. scores of translation component and language model over target data are fused at each character. Augmentation of additional data into a neural encoder-decoder system for solving normalization task has been less explored: we are only aware of the approach reported by Bollmann, Bingel, and Søgaard (2017) who apply lexicon-checking for decoding: during expansion of hypotheses with a beam search guided by cED scores, they remove all characters that would lead to a string not covered by a lexicon.

cED with context signal Linguistic features such as PoS tags, morphological tags and syntactic information, have been used in an attempt to improve the performance of SMT (Koehn and Hoang, 2007), resulting in factored machine translation models. PoS tags and syntactic dependency labels were traditionally used to tackle disambiguation. Sennrich and Haddow (2016) argue that encoder-decoder framework provides a more flexible mechanism for adding linguistic information. In their approach, the embedding layer of an encoder-decoder model with attention is generalized to support the inclusion of additional features by vector concatenation. When words are segmented into sub-word units, the feature value associated with the entire word is copied to all its sub-word units. Similarly, our models incorporate linguistic information in the form of PoS tags, with the difference that an output unit is produced by combining the PoS feature with the current decoder hidden state and soft attention context vector at decoding time. Since we use a character-level model, and thus characters represent our sub-word units, we use the PoS tag of a word for the prediction of each one of its characters.

Hierarchical bi-RNN encoders were initially proposed by Ling et al. (2015a) to improve on language modelling and PoS tagging tasks. Plank, Søgaard, and Goldberg (2016) improve this tagging model further by a) introducing an auxiliary task for word frequency prediction; and b) combining context-level embeddings (higher-level states of the hierarchical encoder) with learned word-level embeddings where the latter are initialized with embeddings pretrained on additional data. Yasunaga, Kasai, and Radev (2018) introduce further improvements to the model of Plank, Søgaard, and Goldberg (2016) (without the auxiliary task) by integrating a conditional random field on top of the higher-level bi-RNN and adapting adversarial training to

the task. Hierarchical bi-RNN encoders were also successfully applied to the task of lemmatization: cED system with hierarchical bi-LSTM encoder of Chakrabarty, Pandit, and Garain (2017) improved results of previous non-neural context-sensitive lemmatizers over 8 languages.

For a range of upstream processing tasks, significant improvements were achieved with a pipeline encoder-decoder systems: one hierarchical bi-RNN is used to first predict a tag (PoS tag or more fine-grained morphological tag), this tag is then fed into another bi-RNN system (not hierarchical) to learn predictions on a final task. Dozat, Qi, and Manning (2017) apply such pipeline approach for dependency parsing. They use a PoS tagging with a tripartite representation for words: character-aware context embeddings (higher-level states of the hierarchical bi-RNN), learned word-level embedding and pretrained word-embeddings. Their systems for tagging and parsing are ranked first in CoNLL 2017 Shared Task on parsing Universal Dependencies (UD) (Zeman et al., 2017). Kanerva, Ginter, and Salakoski (2020) adapt their tagger to predict morphological tag sequences and use it as a part of their pipeline system for lemmatization, achieving state-of-the-art results on UD datasets for 52 languages. Such pipeline approaches differ from our proposed pipeline model cED+HLRNN+Predicted_PoS: we use a simpler architecture where one higher-level bi-RNN component of a hierarchical bi-RNN is used for both tag and, subsequently, normalization prediction. In this way, the most closer to our system is lemmatization model of Kondratyuk et al. (2018). They combine a context-aware cED system augmented with an auxiliary task of predicting morphological tag sequences to achieve improvements over previous solution on lemmatizing morphologically-rich Czech, German, and Arabic. Our system is simpler than their solution in two aspects: we do not employ learned word embeddings (only character-aware context embeddings) and predict a single PoS feature instead of a sequence of morphological tags.

For the task of writing normalization, the employment of context was only studied in the SMT approach of Scherrer and Ljubešić (2016) while we are not aware of such modifications for cED normalization systems. Additionally, while integration of context has been explored in cED systems for other upstream tasks, it has not been studied to which extent two types of the context signal, PoS/morphological tags and signal extracted with a hierarchical bi-RNN from a raw sequence of characters/words, can be mutually substituted and to which extent they are complementary between each other, and towards integration of additional target data. In our experiments, we focus on such analysis by first assessing complementary nature of adding more target data through the synchronized decoding and integrating context signal as PoS tags (Section 5.4). In the next experiment (Section 5.5), we test our context neural systems which can include two types of context signals separately or fuse them together, and identify the complementary aspects of the two types of context encoding.

5.4 Experiment 1: Combining Context as PoS tags with HLLM

To assess how well our methods solve the problems of fluency and ambiguity in text normalization task, we design experiments for a systematic comparison of their performance. We consider two experimental settings, with and without context. As a context model, we choose the system cED+PoS which incorporates context signal in the form of PoS features. This choice allows us to analyze how many of ambiguity cases can be resolved with PoS signal, before moving to more elaborated context signal extraction with hierarchical bi-RNN in Section 5.5. For each setup, we include a setting for synchronized decoding with an additional language model trained on the target side of the data. We evaluate our solution using manually normalized Swiss German corpora: WUS corpus of normalized WhatsApp messages (Stark, Ueberwasser, and Göhring, 2014; Ueberwasser and Stark, 2017) and SMS corpus of normalized SMS messages (Stark, Ueberwasser, and Ruef, 2009-2015).

For the experiments without PoS tags, we run a neural model in two settings: in its plain form of encoder-decoder with attention mechanism (cED) and in combination with an additional language model (cED+HLLMwus+sms). The language model HLLMwus+sms is trained over words using the target side of the two datasets, i.e. the concatenation of the train part of the WUS corpus with the SMS corpus (WUS+SMS).

To evaluate the models with PoS features in a realistic setting, we develop a tagging procedure so that at test time the neural models has only access to the predicted tags. Our basic configuration for the experiments with PoS tags consists of the following steps: 1) training a PoS tagger model using the training portion of the corpus annotated with PoS tags (silver standard, as explained in Section 5.4.2; 2) using the pretrained tagger to predict PoS tags on the development and test portions of the corpus; 3) training a cED+PoS model where PoS tags are used as features. As in the experiments without PoS tags, we use two settings: plain form (cED+PoS) and with advanced decoding (cED+PoS+HLLMwus+sms).

In order to test complementarity of our two proposed modifications to a standard cED, we perform a profound performance analysis. To this end, we consider three classes of words in the test set: NEW, AMBIGUOUS and UNIQUE. The NEW category includes the words that have not been observed in the training set. The other two categories divide the word tokens seen in the training set depending on ambiguity of normalization for a word type. The UNIQUE words are associated with exactly one normalization form in the train set. The last category, AMBIGUOUS, consists of input words which are associated with more than one normalization candidate from the train set. In order to evaluate the difficulty of processing each category, we introduce two baseline models (Section 5.4.3). We expect our modification for the context component to target the category of the AMBIGUOUS words. On the other hand, inclusion of additional target data in the form of HLLM is expected to address fluency by improving normalization of the words in the NEW category.

To further assess the impact of the additional language model alone, we follow the analysis setup for the synchronized decoding (Section 4.4.1) and introduce two subcategories for the NEW class of the input words: we divide the test set token pairs (input word, normalization form) for which the input word has not been seen during training into two classes:

- SEEN_TRG, where the normalization form is a word that has been observed in the target side of the train set;
- NEW_TRG, where the normalization form is a word that has not been observed.

In case of one-to-many or many-to-many alignment units we assign an input word token to NEW_TRG if at least one of the target words in its normalization form is unseen.

We evaluate all the models implemented by reporting word-level accuracy on test set, i.e. we assess whether each test source sequence has been correctly normalized or not by the system. We compute the accuracy of the normalized test set units by comparison with the manual normalization. After performing our quantitative analysis by comparing performance of the considered systems on the different categories of the words, we carry out qualitative analysis where we look more closely into the dataset and analyze performance drops and gains of the neural systems within the categories.

The configuration of the basic component of our model (cED), hyperparameters for training cED and HLLM as well as evaluation approach stay the same as in the previous experiment with the synchronized decoding for the task of canonical segmentation (Section 4.4.1). The only difference is the training time: all neural models are trained for a maximum of 40 epochs.

In the following two subsections, we first introduce details of WUS and SMS corpus, including the procedure for PoS tagging. Then, we discuss the details of the baseline and comparison systems.

5.4.1 Data and PoS Tagging

The data for our experiments comes from manually normalized Swiss German corpora:

- **WUS** set is a corpus of WhatsApp messages (Stark, Ueberwasser, and Göhring, 2014; Ueberwasser and Stark, 2017). The entire collection contains 763,650 messages in different languages spoken in Switzerland. A portion of the data, 5,345 messages in Swiss German, was selected for manual annotation in order to provide a gold standard for automatic normalization. We use this manually annotated portion (a total of 54,202 alignment units) as our main dataset. Table 5.1 in Section 5.2 shows examples of alignment units in the corpus.

- **SMS** set is a corpus of SMS messages, again in different languages spoken in Switzerland (Stark, Ueberwasser, and Ruef, 2009-2015). This is a smaller corpus entirely manually normalized. The Swiss German portion contains 10,674 messages. We use this set (a total of 262,494 alignment units) as training data to train additional language models, as described in more detail below.

All the messages in our dataset are manually normalized using the same web annotation tool and following the same guidelines (Ruef and Ueberwasser, 2013). In our experiments involving models with synchronized decoding component, we integrate the target side of the SMS corpus. In this way, we face a unique setting of employing in-domain extra target data which can be generally expensive to produce.

Our task is to normalize Swiss German WhatsApp messages. In order to train our models, we split the randomly shuffled WUS corpus in 80% training, 10% development and 10% test set, and use this split for all our experiments. The training set contains 43,798 parallel units, the test set 5,043 units, the development set 5,361 units. For the experiments where we train language models with additional target data, we add 262,494 target sequences of the SMS corpus. This results in a total of 306,292 units for the extended target WUS+SMS data.

5.4.2 PoS Tagging

In the following, we describe the procedure for creating a tagged version of the WUS corpus.[2] To train a tagger, we use PoS annotations already provided in the train set. These tags in the train set are produced by an automatic procedure rather than manual annotation, resulting in a silver standard. For completeness, we first report this procedure. Then, we present our approach for training a tagger on the silver standard which we use to tag the development and test set.

Tagging Training Set As a first step, a TreeTagger[3] (Schmid, 1994) parameter file was produced by adapting a general standard German model to the normalized version of the SMS corpus. The general model is adapted by first adding manually to the TreeTagger lexicon all words that were unknown to the initial general model, thus reaching a full coverage. The tagging model is then retrained on the normalized version of the SMS corpus to include Swiss-specific items.

The outcome of the process described above was a TreeTagger parameter file that is specifically tailored to tag normalized Swiss German. This file was used to create a silver standard of the source (not normalized) WUS corpus. First, the normalized forms were tagged with the adapted TreeTagger, which relies on the STTS tagset[4] (consisting of 54 tags), with the additional tag PTKINF for *go*, *goge*, and other forms

[2] The tagged version of the WUS corpus was developed as a part of the collaboration with Massimo Lusetti, Anne Göhring, Tanja Samardžić, Elizabeth Stark and Simone Ueberwasser on the work published in Ruzsics et al. (2019).

[3] http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

[4] http://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/TagSets/stts-table.html

of infinitive particles often encountered in Swiss German dialects. In a second step, the tags were projected from the normalized forms onto the corresponding manually aligned source forms.

Tagging Development and Test Set We trained a tagger model using the BTagger[5] (Gesmundo and Samardžić, 2012) on the train portion of the WUS corpus. In case of one-to-many alignments, where the source form is a contraction of the normalized form, the same PoS sequence refers to both source and target, so it is correct to have one source word that has two or more PoS tags (e.g. source *vorem*, target *vor dem*, PoS APPR+ART). The same applies to many-to-one alignments. Finally, we tagged the source side of the development and test set using the pretrained BTagger.

We have tested the performance of the BTagger by comparing its output to the silver standard, which resulted in 90.30% and 90.67% accuracy on the test and development sets, respectively. These scores, though only an approximation, are higher than those produced by the TreeTagger. Moreover, in our preliminary experiments, we found that using the BTagger also results in a better performance in the normalization experiments which rely on PoS tags. Therefore, we opted for tagging the source side of the WUS development and test set with the BTagger.

5.4.3 Baselines and Comparison

To evaluate improvements with our two main approaches, context signal integration and synchronized decoding, we report results of our basic neural component cED. In the following paragraphs, we present baselines and other systems used for comparison. For our experiments with and without PoS tags we consider separate baselines which are designed to assess the difficulty of the normalization task in the two scenarios. For comparison with neural models in the setting without PoS tags, we also run experiments with a cSMT model, due to its prominent status in the task of Swiss German text normalization.

Baseline. For our baseline in the setting without PoS tags, we adapt an approach which was reported for the normalization task of Swiss German in Samardžić, Scherrer, and Glaser (2015). We consider three classes of input words in the test set: NEW, AMBIGUOUS and UNIQUE, introduced above. For the word in the NEW category, the baseline simply copies such word as its normalization. The UNIQUE words are associated with exactly one normalization form in the train set, which is used by the baseline at test time. For processing the words in the last category, AMBIGUOUS, the baseline uses the most frequent normalization form if there are no ties in their frequencies or, otherwise, randomly chooses a form out of the normalization candidates. The distribution of the three word classes in the test set of the WUS corpus is shown in the left hand side of Table 5.3.

[5] https://github.com/agesmundo/BTagger

Baseline+PoS. In the setting with PoS features we introduce a different baseline, Baseline+PoS. It is similar to the Baseline approach above but it is designed to specifically address ambiguous words. To this end, we consider the same three classes of source words in the test set and normalize the words in the NEW and UNIQUE classes in the same way as it is done by the Baseline. In order to allow Baseline+PoS to use the additional input information in the form of PoS tags, we further split the words in the AMBIGUOUS class into two subclasses. The first subclass consists of words for which each PoS tag that appears together with this word in the train set can be associated with a unique normalization form, i.e. there is a unique normalization for the pair (word, PoS tag). This form is then selected by the Baseline+PoS for the input pair (word, PoS tag) at test time. We refer to this subclass as POS-UNAMBIGUOUS. The other subclass, POS-AMBIGUOUS, consists of the words for which there are at least two normalization forms in the train set associated with the same tag. In such cases, if the tag of the input word has not been observed with this word at train time, the model selects the most frequent normalization of this word in the train set. Otherwise, Baseline+PoS selects the most frequent normalization corresponding to the test input pair (word, PoS tag) or a random form out of its normalization candidates, in case of a tie. The distribution of the word subclasses for the AMBIGUOUS class in the test set of the WUS corpus is shown in the left hand side of Table 5.4.

cSMT. We consider a setting for cSMT with a language model trained over characters using the concatenation of the train part of the WUS corpus with the SMS corpus (cLMwus+sms). Such setting provides a basis of comparison to our cED+HLLMwus+sms model. Note that the cSMT language model operates only at the character level. It is not a trivial task to incorporate a HLLM over words into the cSMT framework and to the best of our knowledge such work has not been done before. We used the Moses toolkit with the following adjustments to the standard settings: i) assuming monotonic character alignment, distortion (reordering) was disabled; ii) in tuning, we used WER[6] instead of BLEU for MERT optimization of the model's components. We used the KenLM language model toolkit (Heafield, 2011) with character 7-grams.

5.4.4 Results and Discussion

The results of our experiments are shown in Table 5.2. In the setting without PoS features, the cED model alone outperforms both Baseline and cSMT. The best accuracy score of 87.09% in this setting is obtained by the cED+HLLM model. This result indicates that the cED approach benefits from the integrated language model for our task.

Turning to the setting with PoS features, the cED+PoS model achieves a substantial improvement over the Baseline+PoS and the best performing model in the setting without PoS (cED+HLLM). Finally, augmenting the cED+PoS model with

[6]WER: Word Error Rate. This metric becomes Character Error Rate in cSMT.

TABLE 5.2: Performance on the task of Swiss German normalization on test set (Word accuracy).

	Setting	Accuracy (%)
Without PoS	Baseline	83.72
	cSMT + cLMwus+sms[a]	86.46
	cED	86.81
	cED + HLLMwus+sms[b]	87.09
With PoS	Baseline + PoS	85.64
	cED + PoS	89.13
	cED + PoS + HLLMwus+sms	**89.53**

[a]cLMwus+sms: character-level language model trained on the target side of the WUS corpus extended with the target side of the SMS corpus.
[b]HLLMwus+sms: higher-level language model trained over words on the target side of the WUS corpus extended with the target side of the SMS corpus.

TABLE 5.3: Performance on the task of Swiss German normalization on test set by source word categories (Word accuracy).

Source words categories			No.	Correct predictions (%)				
				Baseline	cSMT	cED	cED +HLLM	cED +PoS+HLLM
TOTAL			5043	83.72	84.46	86.81	87.09	**89.53**
AMBIGUOUS[a]			1719	80.40	78.94	80.05	80.10	**86.68**
UNIQUE[b]			2714	**98.16**	96.68	**98.16**	**98.16**	97.97
NEW[c]	TOTAL		610	28.85	**62.13**	55.41	57.54	60.00
	WUS[d]	NEW_TRG	364	43.41	x	54.67	x	x
		SEEN_TRG	246	7.32	x	56.50	x	x
	WUS+SMS[e]	NEW_TRG	240	x	57.08	x	47.50	52.50
		SEEN_TRG	370	x	65.41	x	64.05	64.86

[a]AMBIGUOUS: input words with more than one normalization based on the train set.
[b]UNIQUE: input words with one normalization based on the train set.
[c]NEW: input words that have not been seen in the train set.
[d]WUS: only the WUS corpus is used for model training.
[e]WUS + SMS: additional target side of the SMS corpus is used for HLLM training.

TABLE 5.4: Ambiguity resolution analysis with PoS tags.

Source words categories			No.	Correct predictions (%)		
				Baseline +PoS	cED +HLLM	cED +PoS+HLLM
TOTAL			5043	85.64	87.09	**89.53**
AMBIG.[a]	TOTAL		1719	86.04	80.10	**86.68**
	POS-UNAMB.[b]		1071	**91.22**	83.66	**91.22**
	POS-AMB.[c]	TOTAL	648	77.47	74.23	**79.17**
		NEW_POS[d]	11	14.29	14.29	14.29
		TIES[e]	14	57.14	71.43	50.00
		NO_TIES[f]	623	78.97	75.28	80.90

[a] AMBIGUOUS: input words in the test set that have more than one normalization based on the train set.
[b] POS-UNAMBIGUOUS: ambiguous words for which each PoS tag that appears together with this word in the train set can be associated with a unique normalization form, i.e. at test time there is a unique normalization for the input pair (word, PoS tag).
[c] POS-AMBIGUOUS: ambiguous words that have at least two normalization forms in the train set associated with the same tag.
[d] NEW_POS: alignment units (word, PoS tag) in the test set where the word is from the POS-AMBIGUOUS class and the PoS tag is not observed in the train set.
[e] TIES: alignment units (word, PoS tag) where the word is from the POS-AMBIGUOUS class, which are associated with different normalization forms with the same frequencies in the train set.
[f] NO_TIES: alignment units (word, PoS tag) where the word is from the POS-AMBIGUOUS class, which are associated with different normalization forms with different frequencies in the train set.

an additional language model (cED+PoS+HLLM) results in the best overall accuracy of 89.53%.

The results confirm that both approaches for adaption of the plain cED model – synchronized decoding and PoS features – are beneficiary for the task of text normalization. In order to evaluate the two components separately, we assess the performance of our models on the different categories of the input words in the test set. The results for the word categories introduced in the experimental setup, i.e. NEW, AMBIGUOUS and UNIQUE, are presented in Table 5.3. We report the performance of the Baseline, indicating the difficulty of the task for each category, and the best performing models in our two setting, cED+HLLM and cED+PoS+HLLM. To further assess the impact of the additional language model alone, we compare the results of cED+HLLM model to the plain cED model, focusing on the subcategories in the NEW class of the input words, introduced above.

Analyzing the results of the models on the different classes of the input words, we observe that the performance of the cED and cED+HLLM models on the UNIQUE words is identical to that of the Baseline, meaning that they replicate the Baseline strategy for this category. However, there is a slight drop in the performance of the best model, cED+PoS+HLLM, which could be attributed to the higher impact of the PoS features (that could be unseen or noisy) in this model.

The accuracy of all neural models in the NEW category is almost twice as high as the Baseline. This could be explained by the ability of the neural models to

learn well local string transformations, as opposed to the naive copy approach of the Baseline model. The highest score achieved in this category by cED+PoS+HLLM is still relatively low (60%) compared to the performance in the other categories, suggesting that normalization of words not seen during training is a particularly difficult task. Comparing the results of the cED and cED+HLLM models among the three word categories, the highest improvement of around 2 percent points is achieved on the NEW words, resulting in 57.54% accuracy. This suggests that the advanced decoding with additional HLLM particularly helps with the words in this category, which can be explained by the subclass performance. We observe that the HLLM pushes the performance of cED+HLLM higher in the SEEN_TRG subcategory compared to the NEW_TRG subcategory (64.05% vs 47.50%), while the results on these subcategories are relatively similar for the cED model (56.50% vs 54.67%). The difference can be explained by the fact that for the cED+HLLM model the weight of the SEEN_TRG subclass in the NEW category becomes higher due to additional target SMS data used for HLLM training. The synchronized decoding algorithm (optimized for the overall accuracy) drives the HLLM weight up, which results in choosing more normalization forms, out of the candidates generated by cED, that have been observed in the target side. This preference leads to higher performance on SEEN_TRG words, but comes at the expense of a decreased performance on the subcategory NEW_TRG. An additional improvement of 5 percent points is achieved on the NEW_TRG category by cED+PoS+HLLM model. This can be explained by a better ability of this model to learn local string transformations in the presence of PoS features.

The additional PoS features used by the cED+PoS+HLLM model help to improve the accuracy on AMBIGUOUS words by almost 7 percent points compared to the approaches without the PoS tags. We analyze the performance of the PoS-aware models on this category by considering the subclasses of the AMBIGUOUS category introduced for the Baseline+PoS approach: POS-UNAMBIGUOUS and POS-AMBIGUOUS words. In Table 5.4 we show the results on the subclasses, for the best performing model cED+PoS+HLLM, and the baseline model (Baseline+PoS), which gives an estimation of the task complexity in this setting. In order to isolate the impact of the PoS tags in the cED+PoS+HLLM model, we include the cED+HLLM model for comparison.

We observe that while the overall performance of the best model without PoS tags (cED+HLLM) is higher than the Baseline+PoS, its accuracy is inferior on the AMBIGUOUS category and its two subcategories. However, adding PoS tags features is helpful for both classes. In particular, the cED+PoS+HLLM model manages to reach the accuracy of the Baseline+PoS on the POS-UNAMBIGUOUS subcategory, whereas it outperforms the baseline on the subcategory POS-AMBIGUOUS.

5.4.5 Qualitative Analysis

We have quantitatively analyzed the difference in the performance of our systems on three major categories of test input words: NEW, UNIQUE and AMBIGUOUS. In the

following, we perform a qualitative analysis of the performance. We look more closely into the data and analyze what are the typical errors produced by the systems and how the proposed enhancements for the plain cED model – synchronized decoding and PoS features – affect the performance in the different categories.

New words. As already noted in the discussion above, the optimization mechanism in the synchronized decoding used in the cED+HLLM model pushes up the weight of the HLLM component resulting in a higher overall accuracy and, in particular, higher accuracy in the NEW category, i.e. test words that have not been observed in training. We investigate the source of the different performance of cED+HLLM and cED+PoS+HLLM models in the subcategories of the NEW words compared to the cED model (see Table 5.3).

1. *From cED to cED+HLLM: why is there a jump in the performance in the SEEN_TRG category?*

   With the increase of the weight of the HLLM component, more words are normalized by selecting the form out of the cED candidates that has been seen during training. The examples of cED errors which have been corrected with the HLLM in the cED+HLLM system are shown in the Table 5.5. Section a) of the table lists examples of cED errors where the normalization form consists of only one word, i.e. one-to-one alignment units. For example, the word *schi* 'already' was normalized wrongly by cED as *schei*, whereas cED+HLLM picks the right form *schon*, which has been seen in the target side of the train set as a normalization form for other varieties of this input word in Swiss German. In section b) we present examples of the cED errors where the HLLM helps to correct the prediction for the words whose normalization consists of several words, i.e. one-to-many alignment units. This is the category which is specifically targeted by the mechanism of the scores synchronization in the synchronized decoding. For example, cED+HLLM produces the correct normalization form *können wir* for the input word *kömmer* 'we can' while the cED prediction is *kommer*. Finally, with the addition of the SMS target data, more target forms become seen during the HLLM training, which helps the cED+HLLM model to select the right normalization. Some of such examples are presented in section c) of Table 5.5.

2. *From cED to cED+HLLM: why is there a drop in the performance in the NEW_TRG category?*

   While the strategy of increasing the HLLM weight in the synchronized decoding approach helps to improve the overall accuracy score, this comes at the expense of a decreased performance in the NEW_TRG category, i.e. words that have a normalization form which has not been seen in the target side of the train data. For the NEW words which have at least one cED normalization candidate that has been seen during training, the synchronized decoding often results in

TABLE 5.5: Errors of cED in the NEW words category from the SEEN_TRG class corrected by cED+HLLM.

	Input word	Normalization		Eng. transl.	Gold seen	
		cED	cED+HLLM and Gold		in WUS?	in SMS?
a)	schwizer	schwizer	schweizer	Swiss	yes	-
	schi	schei	schon	already	yes	-
	aver	aver	aber	but	yes	-
b)	kömmer	kommer	können wir	we can	yes	-
	hanie	habeie	habe ich	I have	yes	-
	hanise	habe ise	habe ich sie	I have her	yes	-
c)	trurig	trurig	traurig	sad	no	yes
	usfüerige	ausfürigen	ausführungen	executions	no	yes
	gschune	geschune	geschienen	has seemed	no	yes

TABLE 5.6: Errors of cED+HLLM in the NEW words category from the NEW_TRG class corrected by cED.

Input word	Normalization		Eng. transl.	Gold seen	cED+HLLM seen
	cED and Gold	cED+HLLM		in train?	in train?
niveau	niveau	nivea	level	no	yes
essig	essig	essen	vinegar	no	yes
öl	öl	ein	oil	no	yes

selecting this candidate as a prediction. We present some cases where this leads to an error in Table 5.6. For example, the word *essig* 'vinegar' has three cED normalization candidates (sorted by the decreasing cED log-probability score): *essig*, *essen* and *einsig*. While cED correctly normalizes this word as *essig*, cED+HLLM erroneously selects the form *essen* 'to eat' which was observed in the train target data. This kind of errors could be reduced to some extent with the use of more target data for HLLM training. However, due to many rare words according to the Zipf's Law, the HLLM will be overconfident for some cases, no matter how much we increase the training data.

3. *From cED+HLLM to cED+PoS+HLLM: why is there a jump in the performance in the NEW_TRG category?*

   The corrected cases are mostly due to the fact that the PoS features help the cED+PoS model generate better normalization candidates. The synchronized decoding in cED+PoS+HLLM then tends to select the candidate which has been seen in the target training data. This, in turn, leads to the increase in the performance in the NEW_TRG category. To illustrate this case, the word *rumi* 'I clean' has a gold normalization *räume ich* and a silver PoS tag VVFIN+PPER (see Table 5.7). The cED system generates three normalization candidates for this word, sorted by the decreasing cED log-probability score: *rumi*, *rum*

TABLE 5.7: Errors of cED+HLLM in the NEW words category from the NEW_TRG class corrected by cED+PoS+HLLM.

Input	derfür	**rumi**	halt	am	schluss	uf
Pred. PoS	PROAV	VVFIN+PPER	ADV	APPRART	NN	PTKVZ
Silver PoS	PROAV	VVFIN+PPER	ADV	APPRART	NN	PTKVZ
Gold Norm.	dafür	**räume ich**	halt	am	Schluss	auf
Eng. lemma	in return	I clean	just	at	end	up
Eng. transl.	... in return, I will just clean up at the end					
Pred. Norm.:[a]						
cED+HLLM	**rumi**, rum ich, räume					
cED+PoS+HLLM	**räume ich**, rum ich, rume ich					

[a]Pred. Norm.: 3-best predicted normalization forms sorted by the decreasing model score, the best candidate (predicted normalization) is in bold.

ich and *räume*. The first two forms were not observed in the target side of the corpus, whereas the third one was. Contrary to the tendency of the synchronized decoding to pick candidates which have been seen during training (i.e. have a high HLLM score), in this case the cED+HLLM model selects the first form *rumi* as a prediction. This is due to the fact that the cED log-probability for the third candidate is much lower than for the first two and it prevails in the combined weighted cED and HLLM score (i.e., HLLM score and weight are not high enough to select the third option, which was seen in the train set during the decoding). However, with the addition of PoS features, cED+PoS generates a different list of candidates: *rum ich*, *räume ich* and *rume ich* (sorted by the decreasing cED log-probability score). In this case, the weighted combination of the cED+PoS and HLLM scores leads to the selection of the correct candidate *räume ich* by the cED+PoS+HLLM model.

Unique words. We have observed in Table 5.3 that the neural models without PoS features (cED and cED+HLLM) replicate the strategy of the baseline models for the UNIQUE words category by simply generating the normalization form seen during training. However, the accuracy score in this category is under 100% and becomes even lower for the model with PoS features (cED+PoS+HLLM). Next, we present the common patterns for the errors in this category.

1. *Why the performance in the UNIQUE category is under 100% for all models?*

   One of the observed patterns of the mistakes in the UNIQUE category is the wrong inflection ending of an adjective in the normalized form. An example of such error is illustrated in Table 5.8. The input word *nette* 'nice' is associated with the unique normalization *netter* in the training set, which is selected by all the models at test time, although the correct normalization form is *netten*. Taking the context into account could help in such cases. Concretely, recognizing the dative case marker – which is required by the preposition *bei* 'with' – and the feminine marker – suffix *in* of the singular noun *Lehrerin* 'teacher' – in

TABLE 5.8: An example of the errors in the UNIQUE category by the models without PoS features.

Input	bir	**nette**	lehrerin	?
Pred. PoS	APPR+ART	ADJA	NN	?
Silver PoS	APPR+ART	ADJA	NN	?
Gold Norm.	bei der	**netten**	Lehrerin	?
Eng. lemma	with the	nice	teacher	?
Eng. transl.	With the nice teacher?			
Pred. Norm.:				
All models w/o features		**netter**		

the presence of the definite article *der* should result in the adjective *nett* ending with a suffix *en*. Therefore, while the PoS tag alone gives already an indication that the input word is an adjective, more fine-grained morphosyntactic information (or context which can provide this information) is further needed for correct normalization.

2. *Why the performance in the UNIQUE category decreases for the cED+PoS+HLLM model compared to the Baseline and cED+HLLM model?*

   We have found that UNIQUE words which have a wrong predicted PoS tag were particularly prone to be wrongly normalized by the cED+PoS+HLLM model. While the systems without PoS features select the most frequent normalization form for such word in the train set (which almost always leads to the correct prediction), the cED+PoS+HLLM model gives a high weight to the combination of PoS tag and local string transformation. For example, the word *ess* 'eat' is associated with the unique normalization form *esse* in the train set, which is then selected by the cED+HLLM model for a test set example presented in Table 5.9. Moreover, this normalization form is associated with the unique tag VVFIN. This word is wrongly normalized as *ein* by the cED+PoS+HLLM model. The error is caused by the fact that the tag of the word is wrongly predicted as ART instead of VVFIN. The cED+PoS+HLLM model gives a high weight to this PoS signal and normalizes the word as *ein*. This could be explained by a high frequency of normalizing the word *es* as *ein* (indefinite article 'a') in the train data. Therefore, cED+PoS+HLLM gives more weight to the combination of the PoS feature ART and substring *es* and goes beyond the approach of selecting a unique normalization associated with the full input word.

Ambiguous words. As we saw in Table 5.3, the addition of PoS features helps to considerably improve the performance of the systems on AMBIGUOUS words, i.e. test words which have more than one normalization candidate in the train set. The analysis of the performance on the subclasses of the AMBIGUOUS words in Table 5.4 has shown that in almost half of the cases, all the observed input pairs (word, PoS

TABLE 5.9: An example of the errors in the UNIQUE category by the models with PoS features.

Input	**ess**	glich	ez
Pred. PoS	ART	ADJD	ADV
Silver PoS	VVFIN	ADJD	ADV
Gold Norm.	**esse**	trotzdem	jetzt
Eng. lemma	eat	anyways	now
Eng. transl.	I will eat now anyways...		
Pred. Norm.:			
cED+HLLM	**esse**		
cED+PoS+HLLM	**ein**		

tag) for the given word are associated with exactly one normalization form, which is then selected by all the systems with PoS features (POS-UNAMBIGUOUS subcategory). We perform an error analysis of such strategy in this subcategory. In the other half of the cases (POS-AMBIGUOUS subcategory), this strategy is not always applicable since the input word can have the same normalization form associated with different tags in the training. However, the performance of the neural system NM+PoS+HLLM in this subcategory is higher than the Baseline+PoS. We investigate the source of this improvement and the errors in this subcategory of the best performing cED+PoS+HLLM system.

1. *Why the performance of the Baseline+PoS and cED+PoS+HLLM systems on* POS-UNAMBIGUOUS *category is under 100%?*

   Some of the errors in this subcategory come from the incorrectly predicted PoS tag. For example, in the train set the ambigous input word *vor* has been normalized as *vor* ('before'; 'in front of') with the tag APPR and as *von der* ('from the';'of the') with the tag APPR+ART. At test time (Table 5.10), its predicted tag is APPR, while the silver tag is APPR+ART. Therefore, all the models select the incorrect normalization form *vor*.

   In more complicated cases where the tag is correctly predicted, some of the errors come from a further ambiguity in the case markers of the normalized form. To illustrate this case, the ambiguous input word *Lüüt* has been normalized as *Leuten* 'people' with PoS tag NN and *läute* 'to ring' with PoS tag VVFIN. At test time (Table 5.11), the models select the form *Leuten*, corresponding to the (correctly predicted) NN tag. However, the system fails to recognize that the suffix 'n' in the train set is due to the preposition *vor* 'of the', not presented in the test case, which always requires a dative case. Such ambiguity in the case markers could be potentially resolved with the use of context information or a more fine-grained tagset.

2. *Why the performance of cED+PoS+HLLM on* POS-AMBIGUOUS *increases compared to the Baseline+PoS model?*

TABLE 5.10: An example of the errors in the AMBIGUOUS category for the words in the POS-UNAMBIGUOUS class with incorrectly predicted tag by the models with PoS features.

Input	wiu	s	niveau	**vor**	Klass	so
Pred. PoS	KOUS	ART	NN	APPR	NN	ADV
Silver PoS	KOUS	ART	NN	APPR+ART	NN	ADV
Gold Norm.	weil	das	niveau	**von der**	Klasse	so
Eng. lemma	because	the	level	of the	class	so
Eng. transl.	... because the level of the class has been so poor					
Pred. Norm.:						
All systems with PoS				**vor**		
Input	unterirdisch	isch	gsi			
Pred. PoS	ADJD	VAFIN	VAPP			
Silver PoS	ADJD	VAFIN	VAPP			
Gold Norm.	unterirdisch	ist	gewesen			
Eng. lemma	poor	has	been			

TABLE 5.11: An example of the errors in the AMBIGUOUS category for the words in the POS-UNAMBIGUOUS class with correctly predicted tag by the models with PoS features.

Test:					
Input	Pflege	jede	Tag	vier	**Lüüt**
Pred. PoS	NN	PIAT	NN	CARD	NN
Silver PoS	VVFIN	PIAT	NN	CARD	NN
Gold Norm.	pflege	jeden	Tag	vier	**Leute**
Eng. lemma	take care of	every	day	four	people
Eng. transl.	[I] take care of four people every day...				
Pred. Norm.:					
All systems with PoS					**leuten**

Train:							
Input	dass	de	Praktikant	vor	Lüüt	gfluecht	hat
Silver PoS	KOUS	ART	NN	APPR	NN	VVPP	VAFIN
Gold Norm.	dass	der	Praktikant	vor	Leuten	geflucht	hat
Eng. lemma	that	the	intern	in front of	people	curse	has
Eng. transl.	... that the intern has cursed in front of people						

TABLE 5.12: An example of the errors in the AMBIGUOUS category for the words in the POS-AMBIGUOUS class.

Input	i	bi	z	**viel**	gschwumme
Pred. PoS	PPER	VAFIN	APPR	PIAT	NN
Silver PoS	PPER	VAFIN	PTKA	ADV	VVPP
Gold Norm.	ich	bin	zu	**viel**	geschwommen
Eng. lemma	I	have	too	much	swim [pr. perfect]
Eng. transl.	I have swum too much				
Pred. Norm.:					
Baseline+PoS				**viele**	
cED+PoS+HLLM				**viel**	

While the difference in the performance of the two models on the POS-AMBIGUOUS subclass is small, they show an interesting behavior of the cED+PoS+HLLM model. For example, the input word *viel* 'much', with the silver tag ADV at test time, was tagged with the wrong tag PIAT. The word is wrongly normalized as *viele* by the Baseline+PoS model and correctly normalized as *viel* by cED+PoS+HLLM (see Table 5.12). In the train set, it is normalized 26 times as *viel*, with different PoS tags (including 4 times with the tag PIAT and 15 times with the tag ADV) and 6 times as *viele* with the tag PIAT. Therefore, the Baseline+PoS takes the normalization with a higher frequency *viele* for the test input pair (*viel*, PIAT). The behaviour of cED+PoS+HLLM could be explained if we look at the counts of the target forms *viel* and *viele*, not only for the input word *viel* but also for its variants *viil* and *vill*: these forms are normalized as *viele* 97 times and *viel* only 32 times. Therefore, we hypothesize that while the cED+PoS+HLLM model gives a high weight to the PoS features, this weight is balanced with the contribution of the high frequency of the local transformations.

Similarly, the source word *mir* is normalized 78 times in the train set as *mir* ('me' as indirect object) and 82 times as *wir* 'we'. Both normalization forms are personal pronouns and have therefore the same PoS tag (PPER). The Baseline+PoS selects the more frequent form *wir* to normalize the input pair (*mir*, PPER) while the cED+PoS+HLLM model selects again the less frequent form *mir*. This choice could be again hypothetically explained if we look at how many times these two forms were used to normalize dialect variants of the input word *mir* in the train set, such as *mier*, *mer* and others. While the target form *wir* appears in total 144 times, the form *mir* is seen 209 times. Also, the higher frequency of the test cases where the input *mir* is normalized as *mir* is more frequent than the other option. Thus, the choice of the cED+PoS+HLLM model is more advantageous.

3. *Why the performance of cED+PoS+HLLM on* POS-AMBIGUOUS *is under 100%?*

As the previous example of the input word *mir* shows, this word can have different normalization forms which both correspond to the same tag PPER. Another common category of errors is related to the normalization of definite and indefinite articles. They all share the same PoS tag ART, though the normalization forms can be different due to complex morphology, i.e. case, gender and number markers. As previously noted, such errors could be potentially resolved with the use of context information or a more fine-grained tagset.

5.4.6 Conclusion

In our first experiment, we have tested the proposed approaches for the adaption of cED framework to the task of writing normalization: synchronized decoding with an additional word-level language model and integration of context information in the form of PoS tags. In our experiments with Swiss German normalization, we find that two approaches bring complementary improvements of the plain cED system. The decoding approach allows to incorporate more data on the target side and improves performance on unseen input words, while PoS tag signal helps in normalization of ambiguous words. Our qualitative analysis shows that some cases of ambiguity cannot be resolved with PoS features due to the complex morphology of Swiss German. In the next section, we design experiments to test whether ambiguity could be further resolved by integrating a signal from neighbouring words using the hierarchical bi-RNN methods.

5.5 Experiment 2: HLLM with Context as Hierarchical bi-RNNs and/or PoS tags

In the previous section, we have tested how ambiguity resolution in the writing normalization task can be tackled with the inclusion of the context information, in a form of PoS features, into a plain cED system. Our results indicate that some cases of ambiguity require more fine-grained context signal. In this section, we design our second experiment to test whether such more fine-grained signal can be obtained with the hierarchical bi-RNN method alone, or in combination with PoS tags. To this end, we apply our four context models, proposed in the Section 5.2: cED+PoS, cED+HLRNN, cED+HLRNN+PoS, cED+HLRNN+Predicted_PoS. The first two models, cED+PoS and cED+HLRNN, allow a direct comparison of two context methods in terms of overall performance, and in particular, performance on ambiguous words. The third model, cED+HLRNN+PoS, allows to test complementarity of two types of context signal.

In both models with PoS tags, cED+PoS and cED+HLRNN+PoS, we take advantage of the tags already present in a corpus. In the last model, cED+HLRNN+Predicted_PoS, we predict PoS tags using hierarchical bi-RNN component. This method allows to

estimate implications of using such method to jointly learn tagging and normalization. In addition, it allows us to test an alternative to employing corpus PoS tags in the compositional model, cED+HLRNN+PoS. Such tags are costly to produce by manual annotations and often have a silver standard. The latter means that manual annotation is only applied to a portion of a corpus and the rest of the corpus is tagged with a tagger trained on this small manually annotated part. Alternatively, as explained in the previous section (Section 5.4.2) silver standard can be produced by adapting a tagger for a standardized language on the target side of a corpus.

In our experiments, we use Archimob corpus (Samardzic, Scherrer, and Glaser, 2016) for normalization of Swiss German. This choice allows us to perform the following additional analysis. In the experiments with WUS corpus (Section 5.4), we use synchronized decoding to integrate target-side data from SMS corpus. In this way, we included extra in-domain target data which can be generally expensive to produce. As was shown in the experiments with the canonical segmentation task (Section 4.5), integrating noisy target data through our synchronized decoding can lead to improvements too. We test such approach in our current experiment to evaluate its portability to another task. To this end, we run all four context models with an additional setting where we include a HLLM with the synchronized decoding. We choose extra target text data which was already used in another method for normalizing Archimob corpus. Therefore, such setting also allows us a direct comparison to the previous solution.

In the next subsection, we provide details of the experimental setup with respect to data, models used for comparison and neural models implementation.

5.5.1 Experimental Setup

In the following, we first describe Archimob corpus and additional target data employed in the experiments. Then, we discuss our choice of a baseline, models used for comparison and evaluation approach. Finally, we report details of the neural models implementation.

Data For the text normalization experiments we use the ArchiMob corpus[7] which represents German linguistic varieties spoken within the territory of Switzerland. The corpus contains transcriptions of video recordings collected in the context of an oral history project (http://www.archimob.ch) between 1999 and 2001. Currently, the corpus consists of 34 transcriptions of interviews conducted in various Swiss German dialects Samardzic, Scherrer, and Glaser, 2016. In order to reduce intra-speaker and regional variation in transcriptions, and the dialectal variation at the transcription level, each original word form is manually annotated with a normalized form in a subset of 8 recordings which we use in our experiments. Out of 8 documents, 6 are manually annotated with PoS tags while the remaining 2 are tagged with a CRF

[7] https://www.spur.uzh.ch/en/departments/research/textgroup/ArchiMob.html

tagger. The utterances in the corpus are split into syntactically and prosodically motivated segments of 4-8 seconds. We use these segments to extract context information in our experiments with models which integrate context signal through hierarchical bi-RNN. The full dataset (8 documents) is split into train (12,087 segments with 94,122 words), development (1,459 segments with 12,197 words) and test sets (1,055 segments with 8,124 words). For the target-context experiment we use the Standard German OpenSubtitles2016 corpus (Lison, Tiedemann, and Kouylekov, 2018), 185M tokens in size.

Baseline and Comparison We compare our results directly with the current state-of-the art model of Scherrer and Ljubešić (2016). The results for this model were published for a smaller manually annotated portion of Archimob corpus available at that time. We rerun their model on the extended dataset. In order to quantify the difficulty of processing ambiguous words, in the presence of PoS tags, we report results of Baseline+PoS system, introduced in Section 5.4.3. We also report results of our basic neural component cED, in order to evaluate improvements with our two main approaches, context signal integration and synchronized decoding.

Evaluation As evaluation metric, we report word-level normalization accuracy. To evaluate performance of our two main components, context signal integration and synchronized decoding, we follow evaluation setup from Section 5.5 and report results on three categories of input words in the test set: NEW, AMBIGUOUS and UNIQUE. In addition, to analyze the impact of the context models in more details, we report results on two classes of AMBIGUOUS words: POS-AMBIGUOUS and POS-UNAMBIGUOUS, introduced in the context of Baseline+PoS (Section 5.4.3).

Hyperparameters In the neural models with hierarchical bi-RNN component, we use the same size of hidden units in both lower and higher bi-LSTM. Other hyperparameters for training all neural models, HLLM and implementation of synchronized decoding stay the same as in the previous experiment (Section 5.4).

5.5.2 Results and Discussion

The results of the experiments on writing normalization task are shown in Table 5.13. We observe that the context models improve the results of the previous state-of-the-art model from Scherrer and Ljubešić (2016). Adding the HLLM component pushes the results further resulting in 92.43% overall accuracy for the best performing cED+HLRNN+PoS model.

Table 5.14 shows the performance of the models on subcategories of the data. Comparing the results with and without the HLLM (upper and lower half of the Table 5.14) we note that all the models benefit from the additional target data in the category of NEW words resulting in 3-4% improvement within this subcategory across the models. Examining the results for AMBIGUOUS words in the lower half of

TABLE 5.13: Performance on the task of Swiss German normalization on test set for different types of context encoding (Word accuracy).

Setting	Accuracy (%)	
	Without HLLM	With HLLM
Baseline + PoS	83.43	-
cED	88.39	88.69
cED + HLRNN	91.85	92.18
cED + HLRNN + PoS	91.99	**92.43**
cED + HLRNN +Pred_PoS	**92.02**	92.31
cED + PoS	90.10	90.46
cSMT + context[a]	-	89.73

[a] cSMT+CONTEXT: model of Scherrer and Ljubešić (2016)

the table, we note that in comparison to the plain cED system, adding raw context signal (cED+HLRNN) targets PoS-AMB. category while PoS tags (cED+PoS) help with the PoS-UNAMB. words. Overall, there is a stronger preference for the former model due to the dominance of PoS-AMB. subcategory as well as a high weight of AMBIGUOUS words in the data.

Our baseline explains well the difficulty of the task and the behaviour of the cED+PoS model: both reach an accuracy of 88% on the PoS-UNAMB. subcategory. cED+HLRNN improves further the baseline accuracy on the PoS-AMB. category by around 6%. Combining the two context signals in a flat architecture (cED+HLRNN+PoS) brings further small improvements in each category compared to the individual models. It is closely followed by the hierarchical architecture (cED+HLRNN+Predicted_PoS) which interestingly works better on PoS-AMB. We hypothesize that this is due to the presence of non-gold tags in the subset of the data. While predicting PoS tags within the hierarchical model helps on the PoS-UNAMB. category (+1.5% points) compared to the pure context model (cED+HLRNN), it is still lower by 5% points in overall accuracy in comparison to the use of corpus PoS tags in cED+PoS model and the baseline.

5.6 Summary

In this study, we propose a combination of mechanisms for the adaptation of a character-level cED framework to the task of text normalization. The first approach is a variant of the advanced decoding mechanism, introduced in the previous chapter. We reuse it to fuse cED with an additional word-level language model, which allows to incorporate more data on the target side and improve the fluency of the cED output. The second approach is the integration of additional context information in the cED system. For this purpose, we propose several systems which integrate the context signal in two different forms: as PoS tags and through hierarchical language model on the encoder side.

TABLE 5.14: Performance on the task of Swiss German normalization on test set for different types of context encoding by source word categories (Word accuracy).

	No of, %	Baseline +PoS	cED	cED +HLRNN	cED +HLRNN +PoS	cED +HLRNN +Pred_PoS	cED +PoS
Archimob:							
UNIQUE[a]	42.93	98.71	98.68	98.74	**98.80**	98.74	98.60
NEW[b]	10.60	8.13	60.74	**62.14**	61.67	61.32	61.67
AMB.[c]	46.47	86.49	85.19	92.26	92.61	**92.82**	88.74
PoS-UNAMB.[d]	5.3	88.00	79.50	84.00	**89.00**	86.50	87.50
PoS-AMB.[e]	94.7	86.41	85.51	92.73	92.81	**93.17**	88.81
Total		83.43	88.39	91.85	91.99	**92.02**	90.10
+ HLLM:							
UNIQUE	42.93	98.71	98.68	98.74	**98.82**	98.71	98.65
NEW	10.60	8.13	63.30	65.39	**66.09**	64.46	64.92
AMB.	46.47	86.49	85.25	92.24	92.53	**92.74**	88.72
PoS-UNAMB.	5.30	88.00	80.00	84.00	**89.00**	85.50	88.00
PoS-AMB.	94.70	86.41	85.54	92.70	92.73	**93.15**	88.76
Total		83.43	88.69	92.18	**92.43**	92.31	90.46

[a] UNIQUE: input words with one normalization based on the train set.
[b] NEW: input words that have not been seen in the train set.
[c] AMBIGUOUS: input words with more than one normalization based on the train set.
[d] POS-UNAMBIGUOUS: ambiguous words for which each PoS tag that appears together with this word in the train set can be associated with a unique normalization form, i.e. at test time there is a unique normalization for the input pair (word, PoS tag).
[e] POS-AMBIGUOUS: ambiguous words that have at least two normalization forms in the train set associated with the same tag.

In our experiments on Swiss German normalization, we show that both approaches are complementary and result in the improvement of the underlying cED model. In particular, the decoding part helps to improve the performance on unseen input words, whereas context information addresses ambiguous words, i.e. words with different possible normalization forms. We provide a detailed evaluation of the proposed context models and how they target ambiguity. We find that between the two approaches to context encoding, the model with a hierarchical language model results in higher performance improvements over basic cED component and previous solutions than the one with incorporated PoS tags. This improvement is primarily achieved by improving normalization on ambiguous words. Provided a good quality PoS annotation, combining both types of context signal results in further small improvements overall and on ambiguous words, in particular. This result applies to both models proposed to combine two types of signal: in one model, we used corpus PoS tags while in the other one, we used the hierarchical encoder component to predict the tags.

While we test our approaches on the task of Swiss German normalization, the method is highly flexible and conceptually portable to any similar setting of text normalization. In future work, it may be interesting to use our method for the normalization of other languages characterized by dialect variation or non-standard writing.

In addition to our detailed quantitative evaluation, we developed and performed an extensive qualitative analysis of our proposed models. We investigated to which extent our proposed modifications to cED model are sufficient to improve fluency and resolve the ambiguity. In particular, we find that due to several complex morphological phenomena of Swiss German, PoS tags do not provide enough information to resolve some of the cases of ambiguity. In the next chapter, we continue the topic of neural model analysis with a more sophisticated interpretability method which allows to scaling analysis up.

Chapter 6

Multi-level Modelling for Interpretability

Integration of linguistic features into language technologies and analysis of how resulting model performs and handles particular linguistic phenomena has been a common practice in NLP for decades. In the previous chapter, we have demonstrated such study for the task of writing normalization. In the current study, we follow the research line of neural model analysis too. However, we develop more sophisticated methods which allow us to perform an analysis on a larger scale, across many languages.

In this chapter, we concentrate on interpretability methods suitable for examining what knowledge of inflection morphology is captured by neural networks. From a theoretical linguist perspective, such methods are appealing for studying how a particular inflection category is realized in a particular language. For example, what are the ways to express present tense third person plural in Italian? Or in a language, a linguist does not speak? Or a low-resource language with no grammars available? On the other hand, for a computational linguist, inflection is an important component for generating fluent text.

Interpretability research is a vivid topic in the NLP community. This relatively recent research direction in NLP is driven by two major objectives. One goal is to develop analysis methods which accurately represent the reasoning process behind the model prediction. A survey of recently developed methods towards this goal is collected by Jacovi and Goldberg (2020). The increasing need to accurately represent a neural model's decision process stems from ethical considerations of using neural networks in practical applications. Neural networks have been increasingly used in various domains, as different as trading, medicine and government. They are employed for automatic decision-making processes where, traditionally, decisions are taken by a human. This practice showed that neural networks predictions can have biases, for example, towards a specific race or gender. Besides, there is no way to dispute the model's decisions if its reasoning process is not interpretable. This line of interpretability research in NLP driven by ethics is closely connected to the broader interpretability research field in the machine learning community. A comprehensive overview of various desiderata of interpretability research as well as recent research

attempts to conform to them are given in Lipton (2018).

At the same time, there is a growing body of interpretability work in NLP exploring what linguistic knowledge emerges in neural models. In earlier NLP work, the need for profound interpretation methods was less obvious since linguistic properties were incorporated explicitly as features into statistical feature-engineered systems. Such integration has been explored in the neural models too, as we have also demonstrated in the last chapter. However, for many NLP tasks, neural systems are often applied in an end-to-end manner directly on raw sequences of input-output data pairs. This created an increasing need to interpret what linguistic knowledge such models encode. In their survey paper, Belinkov and Glass (2019) categorize recent work on interpreting RNN-based encoder-decoder models, described in Section 3.2, with respect to three dimensions: the kind of linguistic information sought, analysis methods for interpretability, the objects of a neural network under investigations. In more recent work, Manning et al. (2020) propose probing methods for retrieving linguistic knowledge, implicitly learned in a more recent type of encoder-decoder networks with multi-headed self-attention (Vaswani et al., 2017).

Our current work falls into the second category of the interpretability research driven by linguistically motivated objective. In that way, our methodology shares another goal of the interpretability research: methods for linguistic knowledge extraction from neural networks support a cyclical exchange between NLP and fundamental linguistic research. Linguistics research focuses on uncovering patterns and regularities in language. Retrieving and analyzing structures of languages learned by neural agents can systematize our knowledge and, ideally, help us to come up with new regularities. Recent advances (Schrimpf et al., 2020) in testing hypothesis about human language processing using the growing suite of modern interpretable NLP are, indeed, inspiring, but still applied to only few languages. In order to scale up linguistic research, we require truly language-independent models developed for languages other than English (Bender, 2011). Therefore, applying interpretable NLP to language-independent models can aid and scale up linguistic research. In return, understanding the model's decisions can lead to new ideas, how to improve the performance of the model on hard cases, i.e. on a particular linguistic phenomenon or language. In relation to our study, such exchange between two fields naturally emerges too. On the one hand, our methodology is designed to provide tools for linguistic research. On the other hand, aligning extracted inflection patterns learned by the neural network to human reasoning might provide insights on how to improve text generation.

In this study, our goal is to design interpretability methods for neural models which support extraction of inflection patterns in two forms: a) rules expressing morphological patterns for encoding a category and b) data examples realizing this category. Both are illustrated in Ex. 19:

(19) Present Tense 3rd Person Plural: Italian

a. Rules

stem + ***-are*** → *stem* + ***-ano***

b. Examples

comprare → *comprano* '(they) buy'
ballare → *ballano* '(they) dance'
parlare → *parlano* '(they) talk'

where the rule in (19a) can be read as: for verbs in Italian which make up a lemma by combining stem with an affix *-are*, the inflected form for present tense 3rd person plural is formed by concatenating stem with an affix *-ano*.

Traditional sources for studying inflection are limited in terms of providing both rules and examples. Language grammars describe the inflection rules, like the one shown in (19a), but only list a few examples corresponding to the rules. Much more examples can be queried from an annotated corpora. However, the rules have to be deduced from a retrieved list of examples. For example, querying an Italian corpus for the verb tokens in present tense 3rd person plural form would provide a long list of examples, like those shown in (19b). The inflection rules have to be inferred from such list which can be further complicated by the presence of more irregular patterns, i.e. *fare* → *fanno* '(they) do'. Yet in the case of low resourced languages, such resources as grammars and annotated corpora are sometimes scarce or not available.

Another language resource for morphology is typological databases, e.g. World Atlas of Language Structures (WALS) (Dryer and Haspelmath, 2013b). They provide a cross-linguistic categorization of high-level inflection categories but not the rules itself. The categorization is usually formulated in terms of classes of morphological processes which were described in Section 2.1, i.e. suffixation, reduplication and others. For example, particular inflection feature values in Ex. 19 belong to a more broad class of tense-aspect inflection. In WALS, only the categorization of such higher-level category is given: tense-aspect inflection in Italian is formed by suffixation. There are no specific details though describing the suffixation rules in Italian for specific tense-aspect inflection categories, like those listed in Ex. 19.

Contrary to the described limitations of the traditional resources for studying morphological inflection, we propose interpretability methodology for neural models which provides both rules and examples.

The chapter consists of five main parts. In the first part, we introduce the motivation and overview of our approach for interpreting inflection process in neural networks (6.1). The approach comprises two main components: a multi-level neural model trained on the task of inflection generation and a method for morphological pattern extraction from the trained model. In the next two parts, we give a detailed description of our multi-level neural inflection model (6.2) and formalize our pattern extraction method (6.3). In our experiments (6.4), we a) quantitatively and qualitatively analyze extracted morphological patterns; b) evaluate the effectiveness of the proposed multi-level model in terms of performance impact. Finally, we discuss

the strengths and limitations of our methodology in terms of studying inflection phenomena of different typology and propose possible future directions for improving the performance of neural inflection models (6.5).

6.1 Methodology: Interpretability for Inflection Generation

In this section we provide motivation and high-level perspective on our methodology for interpreting inflection generation in neural networks. We begin with grounding an upstream task of inflection generation for extracting inflection knowledge and show limitations for such purposes when using a prevailing class of neural models trained on the task (6.1.1). Then, we give a high-level overview of our methodology - a multi-level modification for a typical neural inflection model and interpretation method for morphological pattern extraction from a trained model (6.1.2). After that, we introduce typologically diverse inflection phenomena which we will use to illustrate our methodology (6.1.3).

6.1.1 Neural Inflection Generation: Opportunities and Limitations for Interpretability

In order to study what inflection knowledge is encoded in neural networks, we choose to analyze neural models trained on the task on inflection generation (introduced in Section 2.2.3). Specifically, we consider a neural model which learns a mapping from a lemma and a sequence of target inflection tags to its inflected form. In this task, the sequence of target inflection tags is usually referred as abstract morpho-syntactic definition (MSD) and comprises a part-of-speech (POS) tag as well as language-specific inflection tags. For example, given the Italian lemma *scolorire* '(they) discolor' and MSD *V;IND;PRS;3;PL* (verb, 3rd person plural present indefinite form), the output is the word form *scoloriscono*:

(20) Inflection Generation Task

Lemma		MSD		Inflected Form
scolorire	+	V;IND;PRS;3;PL	▷	*scoloriscono* '(they) discolor'

Character-level encoder-decoder neural models with attention (introduced in Section 3.2) achieved very high performance on this task across many languages (see Cotterell et al., 2017; Cotterell et al., 2018; Vylomova et al., 2020 for results of recent shared tasks on inflection generation).

Datasets for the task of inflection generation are already available across many languages. Many of such datasets can be extracted directly from inflection tables found in crowdsourced wiktionary data. This is a new and relatively unexplored resource for studying inflection in linguistics.

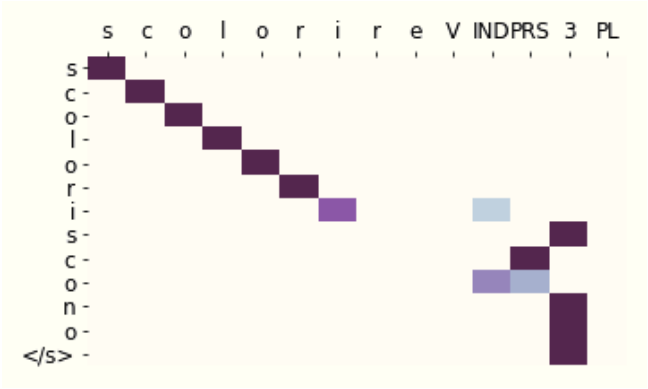

FIGURE 6.1: Example of a heatmap visualizing attention weights learned by model of Peters and Martins (2019). The inflected form is generated from top to bottom.

Both factors, high-performing neural models and multilingual task data, provide a new opportunity to study how a broad set of inflection phenomena is realized cross-linguistically. However, this class of character-level encoder-decoder models for inflection is typically not interpreted: there have been no attempt in the past to extract inflection rules from such neural model.

What allows to reach interpretability in neural inflection models is the *attention mechanism* embedded encoder-decoder architecture (introduced in Section 3.3). Such mechanism serves as an interpretation method on its own. In a character-level model, for a given data example, it provides an explanation which parts of input are important for generating an output character at particular position. This is achieved by extracting a vector of learned attention weights over input positions for each output character position. The higher the weight of an input character, the more 'important' it is to a model for generating a given output position. To find such alignments between input and output positions in a given data example, learned attention weights are usually visualized as heatmaps. An example of an attention heatmap learned on the task of inflection generation is illustrated in Fig. 6.1.

In previous work on inflection generation, attention heatmaps were usually used as a supportive illustration for various attention mechanism proposed. In particular, analyzing attention heatmaps *on selected examples* were used as a supportive evidence that proposed attention models align to a high extent with human intuition. For example, Aharoni and Goldberg (2017) visualize attention heatmaps to demonstrate how their hard monotonic-attention mechanism fit monotonic alignemnt structure found in inflection generation in German. Peters and Martins (2019) apply such methods to particular data examples in two typologically different languages (Azeri and North Frisian) to illustrate how their two-headed sparse attention mechanism handles specific inflection property of exponence.

However, per-example heatmaps provide very limited insight into what a neural agent learns about a specific inflection phenomenon and how neural learning process can be related, in a systematic way, to the linguistic theory. Consider the previous Italian example (Ex. 20): how would humans reason to convert the lemma to its

	-ARE	**-ERE/-IRE**	**-IRE**
1SG	O	O	ISCO
2SG	I	I	ISCI
3SG	A	E	ISCE
1PL	IAMO	IAMO	IAMO
2PL	ATE	ETE/ITE	ITE
3PL	ANO	ONO	ISCONO

TABLE 6.1: Italian verbal inflectional classes, present tense.

inflected form? This is something that we assume that speakers implicitly perform every time they use a word. This is also often an explicit task that language learners perform in the process of acquiring a new language. In this work, we assume that humans apply the rules of grammar (implicit for native speakers, explicit for language learners) when they perform this task. Specifically to our Italian example, human reasoning could look as follows: the verb *scolorire* ends in *-ire* which determines its conjugation patterns (inflection class), illustrated by the third column in Table 6.1. Language learners are often confronted with memorizing this kind of grammatical tables when learning inflection rules. While the table lists the suffixes for making up all present tense forms corresponding to the different values of person and number, our task is to construct a third person plural form. Therefore, we select a suffix *-iscono* at the intersection of the last row ('**3PL**') and the last column ('**-IRE**'). We construct the inflected form by copying the stem *scolor-* and adding the inflection suffix *-iscono*.

What does a typical character-level model of an encoder-decoder class do to perform inflection generation in Ex. 20? In order to interpret model's decisions, we refer back to the attention heatmap of such model on this example in Fig. 6.1. Learned attention weights for generating an output character at a particular position are visualized by a row of corresponding to this position. Such visualization allows to identify most prominent alignments between input and output as cells with the most intensive colouring. By analyzing most prominent alignments, we conclude that model's character-level decisions can be combined into a) copying a substring of characters corresponding to the stem; b) generating characters for a substring corresponding to the inflection suffix. However, the decision why the model chooses a particular inflection class is not visible.

Presence of inflection classes is an important reoccurring pattern in languages. It is one of the most common complications in morphology that morphemes may have allomorphs, i.e. different phonological shapes under different context. Allomoprhy in turn, leads to multiple inflection classes. Formally, inflection class is a set of paradigms that express the same inflectional pattern (we refer to Section 2.1 for definition and examples of paradigms). In our example, Table 6.1 shows a partial paradigm for Italian verb conjugation with three inflection classes. Every row in this paradigm show variation of form, i.e. allomorphy, for a morpheme encoding specific values of number and person in present tense. Languages differ in the number of inflection

classes that they express, number of lexemes belonging to a single inflection class and criteria for assignment of a lexeme to an in inflection class.

In some cases, assignment of lexemes to inflection classes can be expressed explicitly. For example, there are three inflection classes in Italian. Conjugation of verbs for present tense in three classes are illustrated by three columns in Table 6.1. The assignment to the class is determined by an ending in a lemma, indicated in the headers of the columns. There are however further complications to these assignment rules. Verbs ending on *-ire* select between two classes: *-ire* class (third column) is more frequent in terms of types of verbs which belong to it while *-ere* class (second column) is selected by some very frequent verbs ending in *-ire*.

Another type of assignment criteria is phonological: as discussed in Section 2.1, alternations between the forms of phonological allomorphs are often described by morphophonological rules. For example, in Finnish alternations are due to vowel harmony, as illustrated in Ex. 9 which we repeat here:

(21) Finnish (vowel harmony): all suffixes containing one of the harmony vowels */u, a, o, ü, ä, ö/* have two variants, one for back vowels */u, a, o/* and one for front vowels */ü, ä, ö/*. The back form occurs if there is a back vowel to the left, else the front ending variant occurs.

Singular		Plural	
laulaa	'to sing'	*laula-vat*	'(they) sing'
tietää	'to know'	*tietä-vät*	'(they) know'
lukea	'to read'	*luke-vat*	'(they) read'

In some languages, for instance in Tagalog, there is no explicit rule for the inflection class assignment. The term inflection class is not generally used for phonological allomorphy (Haspelmath, 2010). In this work, we will use the terms 'allomorphy' and 'inflection class' interchangeably, i.e. we will refer to all instances of allomorphy as inflection classes, irregardless of whether it is phonologically conditioned or not.

Despite being fairly complicated, allomorphy is an important phenomenon in morphological theory. Therefore, it is expected for an interpretability method to be capable of extracting all patterns of morpheme form variation corresponding to a studied category. In the cases, where assignment of a particular form can be expressed by an explicit rule, it is desirable to make this rule visible in the pattern. We illustrate the expected output of an interpretability method for the previous examples in Italian (Ex. 19) and Finnish (Ex. 21) in the next Example:

(22) Inflection Rules: inflection classes and class assignment

a. Italian *V;IND;PRS;3;PL*

stem + ***-are*** → *stem* + ***-ano***
stem + ***-ere*** → *stem* + ***-ono***
stem + ***-ire*** → *stem* + ***-iscono***

b. Finnish *V;IND;PRS;3;PL*

back-vowel (a,o,u) *stem* → *stem* + ***-vat***
back-vowel (ä,ö,ü) *stem* → *stem* + ***-vät***

To conclude, in order to reach interpretability of models for inflection, we require methods which satisfy three conditions. First, the inflection model's decisions have to be aligned more closely to human reasoning by separating two kinds of operations: determining inflection class versus generating a string given the assignment to a class. Second, systematic analysis of model's decisions requires extraction of inflection rules interpretable to humans. Finally, as we show in the next subsection, both of these factors require working with subword units rather than individual characters, the latter being the prevailing practice for inflection models.

6.1.2 From Character-level Alignments to Subword-level Rules

How to make a character-level neural model for inflection more interpretable? In our work, we tackle the challenge of generalizing from per-example heatmaps (Fig. 6.1) to morphological rules (Ex. 22). We take the stance that, in order to make current models more interpretable, we should analyze their decisions in terms of subwords, i.e. clusters of characters, instead of individual characters. Interpreting character decisions is outside of human intuition due to the double articulation principle (Martinet, 1967). It postulates that single phones are uninterpretable to humans while clusters of them form mental linguistic representation of meaning in the speaker's mind. In writing, this distinction maps to the one between single characters and morphemes, or more general morphological process. The distinctions between morphemes and morphological processes are introduced in Section 2.1.

From this perspective, we propose to extract human-interpretable rules from an encoder-decoder model with attention. Specifically, we modify such model to make it more interpretable by complementing cross-attention mechanism with a novel component (Section 6.2) for self-attention over lemma's subwords. The task of this component is to help identifying the inflection class. To extract the rules, we design a pattern extraction method (Section 6.3) which aggregates learned attention weights a) over a span of characters in a word and b) over a range of words in the same inflection category. To facilitate the use of our methods for linguistic research, we share our code at `https://github.com/tatyana-ruzsics/interpretable-inflection`.

We formulate the following desired properties for retrieving linguistic structures of inflection. For a given grammatical category (e.g. MSD *V;IND;PRS;3;PL* from the previous example in Italian),

P1: (per data example) identify substrings in an inflected form correspondig to morpheme(s) attributed to this category // *scolor**iscono***

P2: (dataset-wide) identify variation in the form of the morpheme (inflection class), e.g. all other substrings attributed to the same category // *-ano, -ono, -iscono*

P3: (per data example) indicate whether triggers for such variation can be attributed to particular substrings of the lemma // *scolor**ire***

P4: (dataset-wide) identify triggers for each inflection class identified in P2 // *-are, -ere, -ire*

Pattern extraction applied to encoder-decoder character-level attention weights retrieves linguistic rules satisfying the requirements P1 and P2, while extracting patterns from subword-level self-attention weights, targets the requirements P3 and P4.

6.1.3 Case studies

In order to demonstrate the use of our approach for linguistic research, we will analyze how well patterns extracted with our proposed methodology align with human knowledge of typologically different phenomena. In morphological typology, cross-linguistic strategies to define form and meaning of morphemes are described by typological parameters (Shopen, 1985; Dryer and Haspelmath, 2013a; Bickel and Nichols, 2007) which separate different dimensions of the strategies. We concentrate on fusion and flexivity dimensions, in order to select typologically different languages for our study.

Fusion classifies how easy it is to find a boundary between a morpheme and its base. It can take the following values: isolating, concatenative and nonlinear. Isolating fusion refers to morphological process where morphemes can appear as separate phonological words. Concatenative fusion describes word formation where morphemes are segmentable from their base, i.e. prefixation and suffixation. Morphological processes are described to have nonlinear fusion when morphemes are not segmentable. The nonlinear fusion, for instance, includes morphological processes of infixation, base modification and reduplication (the description of these processes and corresponding examples are discussed in Section 2.1).

Flexivity parameter describes the triggers for allomorphy. If variation in morpheme form can be explained by phonological processes, such morphological processes are referred to be non-flexive. All other triggers to allomorphy lead to flexive morphological patterns.

We consider verb conjugation rules and select three languages as case studies that cover different degrees of fusion and flexivity: Finnish (concatenative, non-flexive), Italian (concatenative, flexive) and Tagalog (nonlinear, flexive).

Finnish Finnish inflection morphology has concatenative fusion, i.e. morphemes are segmentable. Vowel harmony (Ex. 21) applies throughout all word categories in Finnish. Therefore, verb inflection is formed by non-flexive morphological patterns.

Italian In Italian, morphemes are segmentable too. Italian verbs are conjugated with respect to three inflection classes, defined by lemma endings: *-are*, *-ire*, *-ere* (see Table 6.1). Such flexive form variation is usually described as lexically triggered, i.e. choice of allomorph depends on the individual lexical item.

Tagalog Inflection in Tagalog is formed with affixes, including infixes, and reduplication. Presence of the last two morphological processes implies nonlinear fusion. Form variation of inflection morphemes is flexive. As in Italian, inflection class assignment is lexically triggered but without any explicit rule.

6.2 Neural Model Cross-Att^{ch}Self-Att^{sub}

In this section, we introduce our novel component for **self-attention over subwords** of lemma (**Self-Att^{sub}**). The idea behind this module is to separate model's decisions for each output step generation (guided by cross-attention) from a selection of inflected class which naturally should occur before generation begins. Attention over subwords of lemma is expected to provide decisions for selection of a particular class. This module can be integrated into any variant of encoder-decoder system for inflection with **character-level cross-attention** (**Cross-Att^{ch}**). In this work, we show such integration to a sparse two-headed attention model of Peters and Martins (2019).

Baseline Cross-Att^{ch} (Peters and Martins, 2019) In order to be applied to the task of morphological inflection generation (Ex. 20), the character-level neural model of encoder-decoder type, discussed in Section 3.2, has to be adapted to the twofold input of the task - sequence of characters in lemma and sequence of target inflection tags in MSD. Early models, e.g. Kann and Schütze (2016), simply concatenate the two types of input and encode the resulting string with a bi-directional encoder. Later work discovered that model performance can be improved when treating lemma and tags separately. Ács (2018) proposed to encode both types of input with separate bi-LSTMs and integrate a two-headed attention into decoder: at each prediction step it attends separately to lemma and inflection tags.

To improve interpretability of the inflection model with two-headed attention, Peters and Martins (2019) proposes to combine two attention heads with two mechanisms: sparse activations and gated architecture. Sparse activation function (`sparsemax`), initially introduced by Martins and Astudillo (2016), serves as a sparse alternative to a `softmax` function. The latter yields dense attention weights: all elements in the input always make at least a small contribution to the decision. The gated mechanism provides extra interpretability in the form of a three-way answer about what is relevant at a time step: the lemma, the inflections, or both. This model was among the winners of SIGMORPHON 2019 shared task: ranked first in terms of edit distance.

Due to the high-degree of interpretability of the two-headed gated mechanism, we have chosen the model of Peters and Martins (2019) as a baseline encoder-decoder architecture in our study. In the following, we give a formal description of the baseline architecture. In our formalization, we highlight the main differences between this baseline and the basic encoder-decoder model introduced in Section 3.2.

Input lemma and MSD sequences are represented by separate bidirectional LSTM encoder states: $\mathbf{H}^u$ encodes characters in lemma, and $\mathbf{H}^v$ encodes tags in MSD. The decoder is a unidirectional LSTM. At each prediction step, it computes a hidden state $\mathbf{s}_t$. The attentional decoder state $\tilde{\mathbf{s}}_\mathbf{t}$ is calculated differently than it is done in a standard encoder-decoder (Eq. 3.21). Concretely, at each prediction time t, the model computes two cross-attention heads $\mathbf{u}_t$ and $\mathbf{v}_t$ (Eq. 3.26): one for lemma and one for MSD, by scoring their representations, $\mathbf{H}^u$ and $\mathbf{H}^v$, with a query - decoder state $\mathbf{s}_t$. The alignment scores in Eq. 3.26 are mapped to probabilities using `sparsemax` function.

The two attention heads are used to compute separate candidate attentional decoder states:

$$\tilde{\mathbf{s}}_{tu} = \texttt{tanh}(\mathbf{W}_u[\mathbf{u}_t; \mathbf{s}_t]) \tag{6.1}$$

$$\tilde{\mathbf{s}}_{tv} = \texttt{tanh}(\mathbf{W}_v[\mathbf{v}_t; \mathbf{s}_t]) \tag{6.2}$$

They are combined in a weighted sum to obtain an attentional decoder state $\tilde{\mathbf{s}}_\mathbf{t}$, where weights are calculated by a sparse gate vector $\mathbf{p}_\mathbf{t} = [p_0, p_1] \in \mathbb{R}^2$:

$$\mathbf{p}_t = \texttt{sparsemax}(\mathbf{W}_g[\mathbf{u}_t; \mathbf{v}_t; \mathbf{s}_t]); \tag{6.3}$$

$$\tilde{\mathbf{s}}_t = p_0\tilde{\mathbf{s}}_{tu} + p_1\tilde{\mathbf{s}}_{tv} \tag{6.4}$$

Another deviation from the standard encoder-decoder in Eq. 3.21, is the input-feeding mechanism of the decoder LSTM, initially introduced by Luong, Pham, and Manning (2015). The input-feeding mechanism modifies the type of the input used in the LSTM update (the third line in Eq. 3.21). Instead of using only the embedding of the previous predicted symbol $\hat{\mathbf{y}}_t$ as an input to the decoder LSTM, the input consist of a concatenation $[\hat{\mathbf{y}}_t; \tilde{\mathbf{s}}_t]$ of the predicted symbol and attentional state $\tilde{\mathbf{s}}_t$.

Finally, the activations of the output prediction layer are also sparse, i.e. `softmax` function in Eq. 3.21 is replaced with `sparsemax`. Instead of cross-entropy loss (Eq. 3.24), the model is trained with sparsmax loss (Martins and Astudillo, 2016).

Integrating Self-Att$^{\text{sub}}$ We depart from the existing character-level solution in that we assume that the input to the model - a (lemma, MSD) pair - is complemented with a segmentation of lemma into subwords. To get such extra input, we require integration of subword segmentation into the system. For this purpose, any off-the-shelf segmentation algorithm can be applied. Once the lemma is segmented, the subword boundaries need to be represented in the encoder. We obtain such subword representation of lemma $\mathbf{H}^{subw}$ by first averaging lemma representation vectors in $\mathbf{H}^u$ spanning characters within each subword.

In order to integrate a subword signal into generation, we construct a self-attention head vector $\mathbf{m}$ which is computed once before the decoding stage. It is constructed by scoring the sequence of lemma subword representations in $\mathbf{H}^{subw}$ with a query vector

$\mathbf{q}^{pos}$ corresponding to encoding of lemma's POS tag. This encoding is obtained by selecting a vector in MSD representation $\mathbf{H}^v$ corresponding to the position of the POS tag (e.g, POS tag *V* (verb) is in the first position in MSD *V;IND;PRS;3;PL*).

To integrate subword-level attention head $\mathbf{m}$ into decoding stage, we modify the gate layer in Eq. 6.3:

$$\mathbf{p}_t = \texttt{sparsemax}(\mathbf{W}_g[\mathbf{m}; \mathbf{u}_t; \mathbf{v}_t; \mathbf{s}_t]); \tag{6.5}$$

In this way, the gate mechanism (and decoding) is expected to be informed with a signal for inflection class selection when such signal can be attributed to specific character spans (subwords) in lemma. The attention over subwords is static and shared across target positions, aiming to separate the signal of class assignment it conveys from local transformations, given this assignment.

6.3 Pattern extraction

To extract linguistic rules from the trained model Cross-Att$^{\text{ch}}$Self-Att$^{\text{sub}}$, we summarize its knowledge of inflection into a special database. To populate the database, we analyze predictions of the model on a dataset. This dataset can be the original task data, e.g. a concatenation of train, development and test data. Alternatively, one can used a dataset collected to study a specific inflection phenomenon. For each example in the dataset, we populate the knowledge database with the example itself and two patterns, extracted from the learned attention weights. The first one is a transformation pattern (6.3.1) obtained by applying our pattern extraction method to learned cross-attention weights (Cross-Att$^{\text{ch}}$ component). This method can be applied to any inflection model embedded into encoder-decoder paradigm with attention. The second pattern is over lemma subwords (6.3.2) which is obtained from self-attention weights of the novel Self-Att$^{\text{sub}}$ component. Finally, we explain how the knowledge database, populated in this way, can be queried in order to study inflection phenomena (6.3.3).

6.3.1 Cross-Att$^{\text{ch}}$ Transformation Patterns

The goal of this method is to map each example *(lemma, MSD)* $\rightarrow$ *inflected form* to a transformation pattern of a form $P^{tr}(lemma) \rightarrow P^{tr}$*(inflected form)*. Formally, the input to the algorithm is a lemma $X = x_1, \dots x_n$, MSD $F = f_1 \dots f_l$, predicted target form $Y = y_1 \dots y_m$ and cross-attention weights over lemma characters $A^X = a_1^X \dots a_m^X$, $a_i^X \in \mathbb{R}^n$ and over MSD tags $A^F = a_1^F \dots a_m^F$, $a_i^F \in \mathbb{R}^l$.[1] The output is a string of a form $P^{tr}(X) \rightarrow P^{tr}(Y)$ where constructed pattern representation P^{tr} for

[1] We assume that the sum of weights in a combined vector $[A^X; A^F]$ is 1. In Cross-Att$^{\text{ch}}$Self-Att$^{\text{sub}}$ model this is achieved by scaling cross-attention weights for lemma and MSD with corresponding gate values. Another way, typical for neural inflection models of encoder-decoder class, is to run cross-attention over a concatenation of lemma's characters and MSD's tags.

lemma and target are built through the following steps (illustrated in Table 6.2 for our example in Fig. 6.1):

Input:	
$X = \underset{123456789}{scolorire}$	$F = \underset{1}{V}\ \underset{2}{IND}\ \underset{3}{PRS}\ \underset{4}{3}\ \underset{5}{PL}$
$Y = \underset{123456789101112}{scoloriscono}$	A^X and A^F as in Figure 6.1
Step 1: Transform att. weights into 'salient' alignments	
$A = [\underset{copy}{X_1 \cdots X_7}, F_4, F_3, F_2, F_4, F_4]$	
Step 2: Inverse A and group pred. steps by gen. type	
$X_1 \rightarrow \{c : [1]\}$	$F_4 \rightarrow \{f4 : [8, 11, 12]\}$
$\cdots$	$F_3 \rightarrow \{f3 : [9]\}$
$X_7 \rightarrow \{c : [7]\}$	$F_2 \rightarrow \{f2 : [10]\}$
Step 3: Replace char. in X and Y with indexed gen. type	
$P^{tr}(X) = \underset{7}{\mathrm{c}^1 \cdots \mathrm{c}^1}\ re$	
$P^{tr}(Y) = \underset{7}{\mathrm{c}^1 \cdots \mathrm{c}^1}\ \mathrm{f4}^1\ \mathrm{f3}^1\ \mathrm{f2}^1\ \mathrm{f4}^2\ \mathrm{f4}^2$	
Step 4: Collapse adjacent symbols	
$P^{tr}(X) = \mathrm{c}^1\ re$	
$P^{tr}(Y) = \mathrm{c}^1\ \mathrm{f4}^1\ \mathrm{f3}^1\ \mathrm{f2}^1\ \mathrm{f4}^2$	
Output:	
$\mathrm{c}^1\ re \rightarrow \mathrm{c}^1\ \mathrm{f4}^1\ \mathrm{f3}^1\ \mathrm{f2}^1\ \mathrm{f4}^2$	
($\mathrm{c}^1 \leftrightarrow scolori$, $f4^1 \leftrightarrow s$, $f3^1 \leftrightarrow c$, $f2^1 \leftrightarrow o$, $f4^2 \leftrightarrow no$)	

TABLE 6.2: Illustration of Cross-Att$^{\mathrm{ch}}$ Patterns algorithm applied to the example in Fig. 6.1.

Step 1. Transform input attention weights A^x and A^f into 'salient' alignments A: each component a_j of salient alignments $A = a_1 \dots a_m$ is a set of input positions (in lemma X and/or MSD F) which provide the most significant contributions to predicting a character in Y at position j. We denote positions by capitalized symbols, i.e. $F1$ for position 1 in F, to reflect difference between position's index and value. Salient alignments are built by applying a filtering function ϕ to attention weights at each predicted position: $\phi : [a_j^X; a_j^F] \rightarrow a_j$. In the following, we illustrate how our algorithm works for the simplest choice of the filtering function, max-pooling, which simply selects one input position with a highest attention weight.[2] In our running example, this strategy results in only one element for each component a_j: e.g. $a_7 = X_7$.

Step 2. Inverse mapping A and group prediction steps by generation type: by inverting salient alignments, we construct a mapping from input positions to prediction steps grouped by a symbol corresponding to generation type. The latter is identified for each alignment a_j by the type of input: we denote generation from lemma characters ($a_j = X_i$) by symbol g, while that from a tag ($a_j = F_k$) is denoted

[2] `sparsemax` activations provide another choice to filtering function by keeping the input positions corresponding to non-zero attention weights. In our example, this would result in e.g. $a_7 = \{X_7, F_2\}$. We refer to a more general form of the algorithm covering such case in Appendix A.

by indexed symbol fk. The special case of copying a character from lemma, i.e. $a_j = X_i$ and $x_i = y_j$, is denoted by symbol c. Thus, a position in F can be mapped to only one group of prediction steps (the type of generation is unique and defined by tag's position), while that in X can be mapped to up to two groups, g and c. Some input positions might be absent in the constructed mapping, if not present in salient alignments, e.g. X_9 in Table 6.2.

Step 3. Replace characters in X and Y with indexed generation type symbols: We index (in the order of input positions) triples of salient alignments (input position, generation step, generation type) identified in the previous step. Then, we construct patterns of lemma and inflected form, by replacing characters at aligned positions with an indexed value of the generation type symbol, e.g. $(c, X_j, Y_k) \rightarrow index; x_j \rightarrow c^{index}; y_k \rightarrow c^{index}$. In X, this can result in an aggregated symbol, e.g. replacing X_i with $c^{1;2}; g^1$ means that position X_i is aligned to three target positions, two of which are generated by copying x_i. As illustrated in our running example, we use the same index value in two special cases: a) a whole target substring was copied, and b) a whole target substring was generated by the same tag. We keep the track of symbolic mappings from characters to indexed generation symbols which replace them.

Step 4. Collapse adjacent symbols: Scan representations $P^{tr}(X)$ and $P^{tr}(Y)$, built at the previous step and iteratively collapse adjacent symbols of the same value. In the same time, we update the symbolic mapping: if two adjacent symbols are collapsed, we replace their string mappings with a single mapping from the strings concatenation to the generation symbol.

The idea behind the inverse mapping and indexing in steps 2 and 3 is to ensure a unique way of indexing generation symbols across all data pairs. The indices itself are important to keep one-to-one mapping from substrings to the generation symbols they are replaced with. Both factors come into play when we query the knowledge database for an inflection phenomenon (Section 6.3.3).

6.3.2 Self-Att$^{\text{sub}}$ Lemma Patterns

This algorithm takes as an input a data example (X, Y, F), along with a segmented lemma representation $S(X) = s_1 \dots s_p$ and learned self-attention weights over lemma's subwords: $a^{S(X)} \in \mathbb{R}^p$. The output is a pattern for salient subwords in lemma $P^l(X)$ which is built with a similar procedure as described above where indexing steps 2 and 3 are skipped.

First, we transform self-attention weights $a^{S(X)}$ into salient alignments a by applying a filtering function: $\phi : a^{S(X)} \rightarrow a$, thereby identifying a set of subword positions with the most significant contribution to the overall generation process (any type of filtering function described in the previous subsection can be applied). After that,

we replace all subwords in the input lemma at non-salient positions, $S_j \notin a$, with a dedicated symbol, e.g. asterisk *. Finally, we iteratively merge adjacent asterisk symbols to obtain a more general pattern. To illustrate with our running example, assume that the lemma is segmented into subwords as

$$S(X) = s|col|or|i|re \tag{6.6}$$

We transform learned self-attention weights over subwords $a^{S(X)}$ into salient alignments a by filtering subword positions with non-zero weights. As a result, we obtain two positions, four and five, as salient alignments $a = \{S4, S5\}$. The resulting pattern for salient subwords in lemma is $P^l(X) = *ire$.

6.3.3 Querying Knowledge Database

As a result of applying previous two methods, each data example (X, Y, F), along with segmented lemma representation $S(X)$ and learned attention weights (A^X, A^F, $a^{S(X)}$), can be mapped to two items:

- Cross-Att$^{\text{ch}}$ transformation pattern $P^{tr}(X) \rightarrow P^{tr}(Y)$
- Self-Att$^{\text{sub}}$ pattern for salient subwords in lemma $P^l(X)$

The data examples along with the extracted patterns are stored in a knowledge database. In order to systematically study how the neural model handles a specific linguistic phenomenon of interest, the database can be queried, for patterns and examples, with a phenomenon's formalization in a form of regular expressions applied to lemma, inflected form or MSD. Selected with a query examples are then grouped by their patterns (either transformation or lemma ones) resulting in each group representing an induced linguistic rule for the phenomenon.

At this stage, to make the patterns more readable, we perform unmasking operation within each group: if a particular symbol is used to substitute one substring which is the same for all examples within a group, we replace the symbol back with this substring. For instance, if the pattern from our example $\text{c}^1\ re \rightarrow \text{c}^1\,\text{f4}^1\,\text{f3}^1\,\text{f2}^1\,\text{f4}^2$ represents one such group, and symbol f4^2 is used to substitute only one string *no*, the same across all data points in the group, we can unmask the string, to get a pattern $\text{c}^1\ re \rightarrow \text{c}^1\,\text{f4}^1\,\text{f3}^1\,\text{f2}^1\ no$.

6.4 Experiments and Results

We demonstrate how our framework allows querying morphological patterns learned by an inflection neural model, with three case studies, introduced in Section 6.1.3. The goal of our experiments is to assess how well the extracted patterns correspond to known inflection rules. To see whether our modifications to the inflection model improve its performance, we check the inflection accuracy on the analysed languages and compare it to the original character-level model.

We use data from SIGMORPHON shared task: 2018 edition for Italian and Finnish (10K/1K/1K examples in train/development/test data), and 2020 edition for Tagalog (1,870/236/478). For each language, we train the baseline Self-Att$^{\text{sub}}$ and our model Cross-Att$^{\text{ch}}$Self-Att$^{\text{sub}}$ with batch size 4, beam size 1. Other hyperparameters are kept the same as reported in Peters and Martins (2019). Concretely, a model for each language was trained with early stopping for a maximum of 30 epochs with. We used the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 10^{-3}, which was halved when accuracy on development set failed to improve for three consecutive epochs. Character and tags embedding size is set to 180. LSTMs hidden size is set to 200 for both encoder and decoder. Encoder LSTM for lemma has 2 layers. The dropout for all LSTMs is set to 0.3.

To produce segmented lemma input, we use the Byte Pair Encoding (BPE) method. This unsupervised method for subword tokenization is inspired by data compression algorithm. The latter was originally proposed by Gage (1994) and works by replacing common pairs of consecutive bytes with a byte that does not appear in the data. To perform subword tokenization, BPE method introduced to NLP by Sennrich, Haddow, and Birch (2016) modifies the data compression algorithm of Gage (1994). Concretely, frequently occurring subword pairs are merged together instead of being replaced by another byte to enable compression. It starts by representing each word in a train data as a sequence of characters and proceeds by iteratively merging frequent pairs of characters (or, later, subwords). This process is continued until the desired number of merge operations (a hyperparameter) are completed. In our experiments, we train BPE method for each langauge with 1K merges on a token list (100K examples) extracted from WikipediaDumps articles.[3]

We populate inflection knowledge database using model's predictions on the concatenation of train, development and test set. As a filtering function for constructing salient alignments, we keep only non-zero weights for Self-Att$^{\text{sub}}$ patterns. We choose max-pooling as a filtering function for Cross-Att$^{\text{ch}}$ patterns. We opt for max-pooling instead of keeping non-zero weights since this choice resulted in more general classes of patterns in our initial experiments. This happens due to the fact that on average sparse activations in the attention mechanisms of our model assigned a non-zero weight to only one input feature.

To systematically examine whether the classes of patterns extracted are correct and cover the data adequately, we report two metrics: a) number of examples selected with a query and how many of them are grouped by each pattern; b) model accuracy (correct predictions) with respect to number of examples per selection with a query and per group pattern.

[3] We use archives of the name format enwiki-20190920-pages-articles.xml.bz2 from https://ftp.acc.umu.se/mirror/wikimedia.org/dumps/

6.4.1 Cross-Att^{ch}: Transformation Patterns

For each language, we define specific queries: 3rd person plural present tense for Italian (MSD=*V;IND;PRS;3;PL*), 3rd person plural present tense positive imperative for Finnish (MSD=*V;ACT;PRS;POS;IMP;3;PL*) and imperfective aspect with agent semantic role for Tagalog (MSD=*V;IPFV;AGFOC*). The choice of MSDs is rather arbitrary: for illustrative purposes we select grammatical categories which contain enough examples to represent form variation of the corresponding morpheme.

The extracted Cross-Att^{ch} patterns are presented in Table 6.3 (Finnish), Table 6.4 (Italian) and Table 6.5 (Tagalog). For each query, we show patterns which cover at least 5% of examples selected with query. The patterns are sorted by their number of examples in a decreasing order. For each presented pattern, we show one example mapped to this pattern and symbolic mapping information. We generalize the symbolic mapping across all examples mapped to the pattern in the following way. For each symbol in the pattern, we show all substrings mapped to this symbol along with their frequencies (within a group), if the number of distinct substrings is less than five elements. Otherwise, we show average length ($\approx$) of substrings mapped to this symbol, or exact length ($=$), if it is the same for all of them. These symbolic mappings also include bijection cases ($\leftrightarrow$) which were unmasked after grouping examples (as described in Section 6.3.3).

To illustrate our notations for the generalized symbolic mappings, we concentrate on the first pattern identified for Tagalog in Table 6.5. Here, the mappings are shown explicitly for the three symbols in the pattern, $\mathbf{c^3}$, $\mathbf{c^4}$ and $\mathbf{f3^1}$. Each of them is mapped to one of four different substrings across all the examples grouped by the pattern, i.e. $\mathbf{f3^1}$:{*ag* (131), *a* (8), *ang* (1), *an* (2)}. In parenthesis, the frequency of each substring mapped to $\mathbf{f3^1}$ is given. There is only one case of bijection mapping, $\mathbf{f2^1} \leftrightarrow n$. This means, that in all the examples, the first generated character n is mapped to the same symbol $\mathbf{f2^1}$. Therefore, this symbol is replaced with the string n in the pattern, whereas all the other symbols are kept: $\mathbf{c^{1;2}\ c^{3;4}\ c^5} \rightarrow \mathbf{n\ f3^1\ c^1\ c^3\ c^2\ c^4\ c^5}$. Each of the symbols, $\mathbf{c^1}$, $\mathbf{c^2}$, $\mathbf{c^{3;4}}$ and $\mathbf{c^{1;2}}$, is mapped to a string consisting of exactly one character but there are more than five possibilities for such string. Therefore, only the exact length of the mapping is shown, i.e. $|\mathbf{c^1}|=1$. All the mappings so far show regularities in terms of number of strings or length of strings they cover. As we discuss later, they all represent inflection patterns. This is different for the last mapping, $|\mathbf{c^5}|\approx 3.3$, where symbol $\mathbf{c^5}$ is mapped to variable strings across the examples. Such strings can be identified as root segments.

We observe that in all three cases, the patterns recover inflection morphemes listed in grammars for studied grammatical categories as well as their form variation. For Finnish (Table 6.3), the model correctly identifies morpheme *-koot* as well as its alternation *-kööt*.[4] In addition, the patterns (3) and (4) display morphophonological

[4]For conjugation rules to express imperative form in Finnish, see e.g. https://uusikielemme.fi/finnish-grammar/verbs/verb-tenses-and-moods/third-person-imperative-tulkoon-menkoot-mainittakoon

processes on morphemes boundaries: if a stem ends with *-d* or *-l*, this ending is removed from the final form.

Transformation Patterns	No. of/Acc
Finnish Q: MSD=*V;ACT;PRS;POS;IMP;3;PL*	46/0.91
(1) $\mathbf{c^1\ a} \rightarrow \mathbf{c^1\ k\ oo\ t}$ *karsasta*a : *karsasta*koot $\lvert\mathbf{c^1}\rvert \approx 7.1$; $\mathbf{f6^1} \leftrightarrow k$; $\mathbf{f5^1} \leftrightarrow oo$; $\mathbf{f7^1} \leftrightarrow t$	23/1.00
(2) $\mathbf{c^1\ ä} \rightarrow \mathbf{c^1\ k\ öö\ t}$ *mylvi*ä : *mylvi*kööt $\lvert\mathbf{c^1}\rvert \approx 7.4$; $\mathbf{f6^1} \leftrightarrow k$; $\mathbf{f5^1} \leftrightarrow öö$; $\mathbf{f7^1} \leftrightarrow t$	7/1.00
(3) $\mathbf{c^1\ d\ a} \rightarrow \mathbf{c^1\ k\ oo\ t}$ *promovoi*da : *promovoi*koot $\lvert\mathbf{c^1}\rvert \approx 8.4$; $\mathbf{f6^1} \leftrightarrow k$; $\mathbf{f5^1} \leftrightarrow oo$; $\mathbf{f7^1} \leftrightarrow t$	5/1.00
(4) $\mathbf{c^1\ l\ a} \rightarrow \mathbf{c^1\ k\ oo\ t}$ *aaltoil*la : *aaltoil*koot $\lvert\mathbf{c^1}\rvert \approx 7.7$; $\mathbf{f6^1} \leftrightarrow k$; $\mathbf{f5^1} \leftrightarrow oo$; $\mathbf{f7^1} \leftrightarrow t$	3/1.00

TABLE 6.3: Cross-Att^{ch} transformation patterns for Finnish. *Q* is a query regular expression. Number of examples (*No of*) and accuracy (*Acc*) are shown per selection with query and per group pattern.

In Italian (Table 6.4), the morphemes for all three inflection classes, illustrated earlier in Ex. 22, are present in the patterns: *-ano* (*-are* class) *-ono* (*-ere* class) and *-scono* (*-ire* class). Besides, separation of reflexive ending *si* is visible (pattern (2)) for the *-are* class.

Transformation Patterns	No. of/Acc
Italian Q: MSD=*V;IND;PRS;3;PL*	255/0.99
(1) $\mathbf{c^1\ a\ r\ e} \rightarrow \mathbf{c^1\ a\ n\ o}$ *zampic*are : *zampic*ano $\lvert\mathbf{c^1}\rvert \approx 6.7$; $\mathbf{f3^1} \leftrightarrow a$; $\mathbf{f4^1} \leftrightarrow n$; $\mathbf{f5^1} \leftrightarrow o$	149/1.00
(2) $\mathbf{c^1\ a\ r\ s\ i} \rightarrow \mathbf{si\ c^1\ a\ n\ o}$ *impaper*arsi : *si impaper*ano $\lvert\mathbf{c^1}\rvert \approx 6.8$; $\mathbf{f1^1} \leftrightarrow si$; $\mathbf{f3^1} \leftrightarrow a$; $\mathbf{f4^1} \leftrightarrow n$ $\mathbf{f5^1} \leftrightarrow o$	40/1.00
(3) $\mathbf{c^1\ e\ r\ e} \rightarrow \mathbf{c^1\ o\ n\ o}$ *rirompe*re : *rirompo*no $\lvert\mathbf{c^1}\rvert \approx 7.1$; $\mathbf{f2^1} \leftrightarrow o$; $\mathbf{f4^1} \leftrightarrow n$; $\mathbf{f5^1} \leftrightarrow o$	23/1.00
(4) $\mathbf{c^1\ r\ e} \rightarrow \mathbf{c^1\ s\ c\ o\ n\ o}$ *scolori*re : *scolori*scono $\lvert\mathbf{c^1}\rvert \approx 7.3$; $\mathbf{f4^1} \leftrightarrow s$; $\mathbf{f5^1} \leftrightarrow c$; $\mathbf{f2^1} \leftrightarrow o$ $\mathbf{f4^2} \leftrightarrow n$; $\mathbf{f5^2} \leftrightarrow o$	16/1.00

TABLE 6.4: Cross-Att^{ch} transformation patterns for Italian. *Q* is a query regular expression. Number of examples (*No of*) and accuracy (*Acc*) are shown per selection with query and per group pattern.

Tagalog patterns (Table 6.5) detect two frequent inflection classes for imperfective aspect with agent semantic role. Rules for these classes, corresponding to so-called *um*-verbs and *mag*-verbs, are illustrated in Ex. 23:

(23) Inflection Rules: Tagalog *V;IPFV;AGFOC*

mag-verbs	→	***nagCV*** + *stem*
		(***naglu****luto* 'cooks/is cooking/was cooking')
um-verbs	→	***CumV*** + *stem*
		(***kuma****kain* 'eats/is eating/was eating')

(Schachter and Otanes, 1983)

Each pattern can be decomposed into two components: agent semantic role and imperfective aspect. Agent semantic role expresses the focus of attention in a sentence on the performer of the action. Such information is encoded in Tagalog by prefixation or infixation, with the most frequent affixes being *mag-* and *-um-*. However, there is no explicit rule which verb takes which affix. Imperfective aspect describes events which are not finished yet. Aspects in Tagalog are not associated with a tense system. Each aspect can be translated into English using several tense-aspect formations in English. For example, imperfective aspect corresponds to present simple, present continuous or past continuous tense. Imperfective aspect in Tagalog is formed by reduplication which has different forms for focus prefixes and infixes, as illustrated in Ex. 23. In addition, focus prefix *mag-* changes its form to *nag-*.

Analyzing symbolic mappings in Tagalog patterns, we conclude that the pattern (1) encodes prefixation (the most frequent prefix in this group is *nag*) with subsequent copying of the first syllable and copying of full lemma string. The pattern (2) expresses reduplication of the first consonant (which interestingly gets aligned to a tag rather than to the first character of the lemma), generating infix (the most frequent infix in this group is *um*), copying the second character of the lemma (vowel, as seen from the mapping statistics), and then copying of full lemma string.

6.4.2 Self-Att^{sub}: Lemma Patterns

To see whether our model uses indeed subword segments when choosing the specific variant of a morpheme, we analyse Self-Att^{sub} lemma patterns. Concretely, we use as queries regular expressions on MSD and the target form to select examples corresponding to a specific inflection class, identified above with transformation patterns. When assignment to such class can be attributed to specific subwords in lemma, we expect to find frequent lemma patterns in the examples selected by the query, where such subwords are identified as salient.

The queries and extracted patterns for Italian and Finnish are presented in Table 6.6 and Table 6.7 correspondingly. We note that for Tagalog, we do not find any frequent patterns. This finding is in line with no explicit criteria for inflection class assignment in this language. For instance, there are no clusters of lemma patterns

Transformation Patterns	No. of/Acc
Tagalog	
Q: MSD=*V;IPFV;AGFOC*	377/0.86
(1) $\mathbf{c^{1;2}}$ $\mathbf{c^{3;4}}$ $\mathbf{c^5}$ → **n f3^1** $\mathbf{c^1}$ $\mathbf{c^3}$ $\mathbf{c^2}$ $\mathbf{c^4}$ $\mathbf{c^5}$ *paalam* : *nagpapaalam* |$\mathbf{c^1}$|=1; |$\mathbf{c^2}$|=1; |$\mathbf{c^{1;2}}$|=1; |$\mathbf{c^{3;4}}$|=1 $\mathbf{c^3}$: {*a* (82), *u* (34), *i* (23), *e* (3)} $\mathbf{c^4}$: {*a* (82), *u* (34), *i* (23), *e* (3)} **f3^1**:{*ag* (131), *a* (8), *ang* (1), *an* (2)} **f2^1** ↔ *n*; |$\mathbf{c^5}$|≈ 3.3	142/0.90
(2) $\mathbf{c^1}$ $\mathbf{c^{2;3}}$ $\mathbf{c^4}$ → **f2^1 f3^1** $\mathbf{c^2}$ $\mathbf{c^1}$ $\mathbf{c^3}$ $\mathbf{c^4}$ *hiram* : *humihiram* |**f2^1**|=1; |$\mathbf{c^1}$|=1;|$\mathbf{c^{2;3}}$|=1; |$\mathbf{c^4}$|≈ 3.1 **f3^1**: {*um* (71), *am* (3), *k* (1), *as* (1), *an* (2)} $\mathbf{c^2}$: {*i* (24), *a* (35), *u* (17), *o* (2)} $\mathbf{c^3}$: {*i* (24), *a* (35), *u* (17), *o* (2)}	78/0.86

TABLE 6.5: Cross-Att^{ch} transformation patterns for Tagalog. *Q* is a query regular expression. Number of examples (*No of*) and accuracy (*Acc*) are shown per selection with query and per group pattern.

for a query 'gold target=*nag** & MSD=*V;IPFV;AGFOC*'). Such query would filter lemma patterns corresponding to the words belonging to one of the inflection classes identified earlier with transformation patterns.

For each query, we list the most frequent patterns (sorted by frequency in a decreasing order) along with one segmented lemma example mapped to the pattern.[5] The segments of lemma examples, identified as salient (and presented in the patterns) are highlighted in bold.

We conclude that the subword regions identified by Self-Att^{sub} patterns conform to a great extent to triggers of morpheme form variation listed in grammars. We note that while the regions for finding such clues (when they are phonological or lexical, and frequent) look plausible, their form is influenced by the results of BPE segmentation and may be not perfectly aligned with grammars. For example, Italian patterns (Table 6.6) show that the model's focus is on the endings of lemmas for all three classes, listed in the grammar rules (Ex. 22). In case of reflexive verbs, where reflexive ending *-si* tends to be separated into a separate subword by BPE, the model correctly places focus on a more informative penultimate segment.

The patterns extracted for Finnish (Table 6.7), display the grammar rules too: the focus on the lemma endings *-aa/-ua* for the first group, and *-ää/-ä* for the second group, points directly to the harmony of back and front vowels (Ex. 22), respectively. The model does not search for the clues in vowel patterns of stem but chooses a smart strategy to focus directly on the inflection endings for lemmas: they are frequent and already agree with the vowels found elsewhere in the stem to the left.

[5] We refer to Appendix B, Tables B.1-B.2 for the full list of the extracted patterns.

Lemma Patterns	No. of/Acc
Italian	
Q: gold target=*scono & MSD=msd_it	23/1.00
re** (*in*| *z*| *o*| *ti*|*chi*| ***re)	9/1.00
ire** (*s*| *col*| *or*| ***ire)	7/1.00
ir (*re*| *in*| *ser*| ***ir***| *si*)	6/1.00
Q: gold target=*ano & MSD=msd_it	189/1.00
are** (*z*| *am*| *pic*| ***are)	149/1.00
arsi** (*im*| *pa*| *per*| ***arsi)	26/1.00
car (*ri*| *mb*| *ec*| ***car***| *si*)	3/1.00
Q: gold target=*ono & !(*scono) & MSD=msd_it	41/0.95
ere** (*ri*| *otten*| ***ere)	19/0.95
dere** (*te*| *le*| *ve*| ***dere)	10/1.0
ger (*cos*| *par*| ***ger***| *si*)	3/1.00

TABLE 6.6: Self-Att^{sub} Patterns for Italian. *Q* is a query regular expression, *msd_fin* is *V;ACT;PRS;POS;IMP;3;PL*, *msd_it* is *V;IND;PRS;3;PL*. Number of examples (*No. of*) and accuracy (*Acc*) are shown per selection with query and per group pattern.

Lemma Patterns	No. of/Acc
Finnish	
Q: gold target=*koot & MSD=msd_fin	37/0.97
aa** (*kar*/*sa*/*st*/aa***)	8/1.00
ua** (*ku*/*or*/*ett*/ua***)	5/1.00
Q: gold target=*kööt & MSD=msd_fin	9/1.00
ää** (*jä*/*n*/*ist*/ää***)	3/1.00
ä** (*v*/*et*/*ele*/*hti*/ä***)	3/1.00

TABLE 6.7: Self-Att^{sub} Lemma Patterns for Finnish. *Q* is a query regular expression, *msd_fin* is *V;ACT;PRS;POS;IMP;3;PL*, *msd_it* is *V;IND;PRS;3;PL*. Number of examples (*No. of*) and accuracy (*Acc*) are shown per selection with query and per group pattern.

6.4.3 Self-Att^{sub}: Performance Impact

We evaluate the impact of the novel Self-Att^{sub} component by comparing our multi-level model, Cross-Att^{ch}Self-Att^{sub} , with the baseline model, Cross-Att^{ch} , in terms of two factors: performance on test set and number of trained parameters. For reference, we include results of

- hard monotonic attention (HMA) system of Wu and Cotterell (2019) that currently holds the state-of-the-art on the inflection generation task. We rerun their code on our datasets.
- a variant of our system, Cross-Att^{ch}Self-Att^{ch} , where the self-attention module is run over characters of lemma, instead of subwords. Such choice corresponds to a limiting case of lemma segmentation where each character is a segment.

We report accuracy and edit distance on the test set as well as the number of trained parameters for each model in Table 6.8. The number of parameters for Cross-Att^{ch}Self-Att^{sub} model is the same as for its character variant Cross-Att^{ch}Self-Att^{ch} . The difference in the number of parameters across the languages is due to variation of their character vocabulary sizes.

	Baseline	Our model	Comparison	
	Cross-Att^{ch}	Cross-Att^{ch}Self-Att^{sub}	Cross-Att^{ch}Self-Att^{ch}	HMA
Italian	95.40	96.70	**97.40**	96.80
	(0.09)	(0.09)	**(0.06)**	(0.29)
	[1,742K]	[1,783K]	[1,783K]	[8,647K]
Finnish	93.80	**94.40**	93.60	93.90
	(0.13)	(**0.09**)	(0.12)	(0.13)
	[1,758K]	[1,798K]	[1,798K]	[8,709K]
Tagalog	65.75	**69.98**	66.81	63.39
	(1.21)	(**0.92**)	(1.00)	(1.53)
	[1,739K]	[1,780K]	[1,780K]	[8,623K]

TABLE 6.8: Performance on the task of morphological inflection generation. Accuracy (and edit distance) on test set. The number of trained model parameters is given in squared brackets.

We observe that the Cross-Att^{ch}Self-Att^{sub} model shows systematic improvements across all three languages over the baseline and reference models. With regard to the level of segmentation, the character variant of our system, Cross-Att^{ch}Self-Att^{ch} , achieves higher results on Italian. Such result can be explained by the inflection class assignment criteria in Italian (Table 6.1). The words are associated with their inflection class depending on a certain vowel (*a*, *e* or *i*) in a certain position in the lemma. As discussed before, grammars usually describe these rules in terms of lemma inflection endings (*-are*, *-ere* or *-ire*). However, as we have seen in the extracted lemma patters (Section 6.4.2), subword segmentation methods do not always produce the segments of such form. For example, *telvedere* is segmented as *te*| *le*| *ve*| *dere* (Table 6.6). We hypothesize that in the limiting case of segmentation used in the model Cross-Att^{ch}Self-Att^{ch} , the segments become more frequent and therefore, provide more useful signal for inflection class assignment.

In terms of the number of trained parameters, the improvements due to Self-Att^{sub} component are achieved by only adding a relatively small number of extra parameters compared to the baseline model, Cross-Att^{ch} . We also note that the performance of our systems is higher or on par with the state-of-the art model HMA, while the latter has on average seven-fold increase in the number of parameters, compared to that of Cross-Att^{ch}Self-Att^{sub} and Cross-Att^{ch}Self-Att^{ch} .

6.5 Discussion

In the following, we discuss our proposed methodology and future work in terms of two aspects, interpretability and performance. First, we analyze the range of

morphological patterns which are possible to extract with our methodology. Then, we propose future directions for improving performance of neural inflection models cross-linguistically.

6.5.1 Interpretability

In the following, we reflect on what kind of inflection phenomena, it is possible to study with our proposed framework. We list the range of inflection patterns which can be extracted with our methods in terms of typological parameters.

Fusion In the experiments, we illustrate that our Cross-Att^{ch} transformation pattern approach can effectively extract rules for concatenative morphological patterns as well as reduplication processes. What is beyond, at the moment, are nonlinear processes which are not always visible in orthography, e.g. tonal changes and internal stem changes. The latter, for example, is demonstrated by root and pattern morphology in Arabic and Hebrew, for which standard orthographies do not indicate most vowels.

Flexivity Our Self-Att^{sub} lemma pattern method is able to identify phonological (visible in orthography) as well as lexical triggers to variation of inflection morpheme's form. However, the case of irregular forms which apply to only a few number of lexemes (e.g. English *go*→ *went*) would not be identifiable in patterns. While allomorphy cases like this are likely to be fairly rare in terms of word types, they seem to be only maintained in high-frequency words (Bybee, 1985). Thus, although affecting only a small number of words, irregular forms might be visible in patterns when studied together with word frequency. At the moment, such analysis is not possible due to the current practices of building inflection generation datasets: while inclusion of data examples into datasets takes into account frequency distribution of inflected forms (Cotterell et al., 2017), the frequency information is not present in the dataset.

Exponence This parameter encodes the extent to which single morphemes express multiple morphosyntactic features. Morphemes can exhibit monoexponence versus polyexponence. For the class of neural models currently used for inflection generation, it is not possible to see a clear correspondence between the meaning assigned by humans and the model: as we see from Fig. 6.1 which illustrates polyexponence in Italian inflection, the model assigns separate characters of inflection morpheme *-scono* to different tags while to human, it is hard to break down this morpheme into smaller meaningful parts. This challenge is exemplified by the input features identity problem in attention weights (Brunner et al., 2019).

6.5.2 Performance

Future work can evaluate an impact of the novel subword-level self-attention component Self-Att^{sub} in combination with frequently applied induction biases to the inflection generation task. For example, hard monotonic attention used in HMA system of Wu and Cotterell (2019), proved to be highly effective for indo-european languages with concatentive morphology. The model holds state-of-the-art result on SIGMORPHON 2017 dataset where most of the languages are from indo-european family. The last instantiating of the shared task (Vylomova et al., 2020) for typologically-distinct languages reinforced this result. The results of this shared task also showed that for the languages with non-concatenative morphology, the models with hard monotonic attention have lower performance than encoder-decoder models of transformer paradigm (Vaswani et al., 2017). Integrating our novel self-attention over subwords into character-level transformer model is another promising future direction.

6.6 Summary

Neural models for morphological inflection recently achieved very high results, but their interpretation remains challenging. Towards this goal, we propose a simple linguistically-motivated variant to the encoder-decoder model with attention. In our multi-level model, the character-level cross-attention mechanism is complemented with a self-attention module over the input substrings. In order to interpret what the model learns, we design a novel approach for pattern extraction from attention weights. We apply our methodology to analyze model's decisions on three typologically-different languages and find that a) our pattern extraction method applied to cross-attention weights uncovers variation in the form of inflection morphemes; b) pattern extraction from self-attention shows triggers for such variation; c) both types of patterns are closely aligned with grammar inflection classes and class assignment criteria, for all three languages. In terms of performance impact, we find that the proposed self-attention component leads to consistent performance improvements over a strong baseline. Our multi-level model performs better or at par with the previous solutions while using fewer trainable parameters. Our experiments with linguistic rules induction illustrate the great potential of our methodology for linguistic research scaled to languages of diverse typology. It allows employment of resources which are not traditionally used for theoretical studies of inflection: crowdsourced wiktionary data and high-performing neural inflection models.

Chapter 7

Conclusion and Outlook

In this thesis, we presented original multi-level modelling for upstream text processing. Our methodology is developed to achieve several goals. First of all, our objective is to improve the **performance** of the models for three upstream processing within the character-level RNN encoder-decoder framework. We focus on developing methods which are **portable** across tasks and languages, to the extent where it is possible given a high degree of structural variability in languages discussed in the introduction. Finally, our methods aim to achieve a higher degree of **interpretability** in neural modelling. Such objectives allow for the methods which scale up language technologies to more and more languages and, at the same time, provide new resources and opportunities to test theories in theoretical linguistics.

7.1 Contributions

All our proposed architectures are **hierarchical**. They integrate higher-level signal into character sequence transformation modelled with a character-level encoder-decoder system (cED). The higher-level signal refers to information which can be extracted from units higher than characters. We considered two types of higher-level signal in our models. Sequential subword signal is extracted from segmenting processed words into subwords. In different upstream tasks, these subwords can correspond to morphemes, substrings or even individual words. All our architectures which integrate a subword signal are novel and improve over previous solutions. The second type of higher-level signal refers to the use of words in context: we extract and incorporate context information in the form of PoS tags and surrounding word sequences within an utterance. While the architectures we propose to use for context-level integration are not new in itself, they have not been adapted before for the upstream tasks we consider in this thesis.

The hierarchical components which we propose are linguistically motivated by double articulation principle (Martinet, 1967) and allow us to make a methodological contribution to upstream processing in light of the objectives listed above.

Our first novel methodological innovation is a **synchronized decoding algorithm**. The algorithm modifies beam search in a standard character-level encoder-decoder system. It allows integration of an additional component, a language model

trained over higher-level units, i.e. segments, on a target side. The scores of two components are combined in the decoding stage at synchronization points which correspond to segment boundaries. We showed that such multi-level model improves performance on the task of canonical morphological segmentation. We found that it is especially suitable for morphologically rich languages with regular concatenative patterns, e.g. Indonesian. While we developed our multi-level architecture with synchronized decoding for the task of word segmentation, we argued that it could be adapted to other upstream tasks where segmentation is more implicit. We showed such portability to the task of Swiss German writing normalization. The implicit segments in Swiss German words manifest themselves as separate words in Standard German after performing normalization. Although the multi-level model with synchronized decoding was designed to learn from a small parallel corpus, our approach is flexible towards the integration of more target data. Additional target data specifically developed for a task at hand is an expensive resource. We showed that additional improvements are possible to achieve by integrating heterogeneous language data. For the task of canonical segmentation, integrating dictionaries leads to further improvements on unseen input words whose segmentation contains segments unseen in training data, especially new roots. For the task of writing normalization, integrating text data in canonical language on the target side leads to improvements in normalizing unseen words.

As our second methodological contribution, we presented a multi-level model for solving the task of writing normalization. In our model, we propose two modifications to a standard cED system which **combine two types of higher-level signals**: **word-level** signal on the target side of data and **context** signal on the source side. We adapted previous solutions for extracting and integrating these two types of information. We trained a language model over words on the target side for the word-level signal, which we integrated by adapting the synchronized decoding. We proposed and tested several solutions for integrating context: as PoS tags, as an additional hierarchical language model on the encoder side, as well as a combination of the two. We developed and carried out an extensive quantitative and qualitative evaluation analysis of applying our approach to the task of Swiss German normalization. This analysis showed a) all our models improve over previous solutions; b) complementarity of our two proposed modifications; c) most suitable settings for the context integration given the morphological properties of Swiss German. Our two modifications' complementarity was shown by analyzing the model performance on the different categories of test words. We found that the target-side language model helps to improve the performance on unseen input words, whereas context information addresses ambiguous words, i.e. words with different possible normalization forms. We showed that certain morphological phenomena in Swiss German do not allow PoS information to resolve ambiguity in normalization fully. Comparing two types of context encoding, we showed that integrating context signal with a hierarchical language model along results in better ambiguity resolution and greater overall improvements over the basic

cED component. Combining both types of the proposed context encoding – PoS tags and encoder-side hierarchical language model – results in the further improvements on ambiguous words, provided a good quality PoS annotation.

Finally, we proposed a novel **interpretability methodology for extracting knowledge of inflection morphology** learned by neural network models. Our methodology consists of **a novel multi-level model** for the task of morphological inflection generation and **an interpretation method** to analyze what such model learns. First, we provided a linguistic motivation why interpreting character-level decisions of a cED model for inflection generation does not align well with human intuition and why such models should include subword-level information to achieve higher interpretability degree. We then showed how to integrate subword-level signal into a cED system by combining cross-attention with **a self-attention component over subwords**. Our interpretation method was designed to extract linguistic rules from the learned alignment weights of the two attention components. We designed experiments to test our methodology on three typologically-different languages in terms of several aspects: 1) performance impact of a novel subword-level components; 2) interpretability degree of our pattern extraction method in capturing inflection phenomena of different typology. We showed that the proposed self-attention component leads to consistent performance improvements over a strong baseline. Our model performs better or at par with the previous solutions while using less trained parameters. We found that a) our pattern extraction method applied to cross-attention weights uncovers variation in form of inflection morphemes; b) pattern extraction from self-attention shows triggers for such variation; c) both types of patterns are closely aligned with grammar inflection classes and class assignment criteria, for all three languages.

Our multi-level solutions for improving upstream processing can support the development of downstream applications and provide new material for aiding fundamental linguistic research. We believe that our ideas for linguistically motivated multi-level modelling would stimulate their use in developing high-performing, portable and interpretable NLP models in the future.

7.2 Future directions

In the following, we point to the possible future directions when developing hierarchical models for upstream processing.

In light of the recent developments in NLP, there has been a shift in the encoder-decoder paradigm from RNN-based architectures, considered in this thesis, to transformer encoder-decoder systems (Vaswani et al., 2017). While these developments made transformer architecture ubiquitous in downstream systems, upstream processing modelling has not followed quite yet. Only recently, there appeared first attempts to adapt a transformer system to character-level transduction. For example,

a character-level transformer was recently proposed as one of the neural baseline models in the recent shared task on morphological inflection (Vylomova et al., 2020). We believe that our approaches for higher-level signal integration, synchronized decoding (Chapter 4) and static self-attention over subwords (Chapter 6), are highly portable as it is to the character-level transformer. Our models for context integration (Chapter 5) are more specific to RNN architecture. However, the idea of running encoder on different levels of the input could be adapted to the transformer architecture too. Such hierarchical systems could be explored to improve further on the upstream tasks.

In our experiments with context integration into cED, tested in Chapter 5 on writing normalization task, we showed that the most benefits are achieved when combining two types of context encoding: as a hierarchical language model on the encoder side and PoS tags. In this system, PoS tags are predicted at the test time using a composite character-level and word-level representation of the input word, both coming from the hierarchical language model. One could explore combining such representation with pretrained word embeddings. Such tripartite representations for PoS tag prediction has been used in the winning system for tagging and parsing in CoNLL 2017 Shared Task on parsing Universal Dependencies (Zeman et al., 2017). It was also later adapted in a state-of-the-art model on another upstream task of lemmatization (Kanerva, Ginter, and Salakoski, 2020), where tagging is integrated into lemmatization as a pipeline approach. Another possible direction for integrating context into cED is by replacing word-level representations with subword-level ones. This would result in more informative regular segment patterns that could help cED system improve tagging and ambiguity resolution in the task. One more idea for integrating more regular patterns into the task of writing normalization is to model it on the level of utterance. In contrast to our approach of integrating the context by exploring the utterance on the source-side only, the target-side context could provide more regular patterns since variation in writing is reduced in canonical language through normalization.

While we were guided with portability objective in developing our methods, they could be more specifically adapted by integrating task-specific and language-specific induction biases. For example, in the task of inflection generation, a number of models have successfully explored the integration of hard monotonic attention into the cED system (Aharoni and Goldberg, 2017; Makarov, Ruzsics, and Clematide, 2017; Wu and Cotterell, 2019). Such mechanism has been shown to work remarkably well in the languages with concatenative morphological patterns.

Finally, while our interpretability methodology (Chapter 6) was proposed to analyze models for inflection generation, we believe that its general idea is transferable to analyze neural models with attention in other domains. Specifically, analysis of transformation patterns averaged over attention weights and data examples could be adapted to other tasks. Such methods would provide an alternative to prevalent per-example heatmaps and display a more general picture of what a neural model learns.

Appendix A

Cross-Att^{ch} Transformation Patterns Algorithm

In this Section, we formalize Steps 2 and 3 of the algorithm for pattern extraction from character-level decoder attention weights presented in Section 6.4.1.

Algorithm 4: (Step 2) Inverse salient alignments mapping A and group prediction steps by generation type

Inputs:
$X \leftarrow [x_1 \dots x_n]$; // Lemma
$F \leftarrow [f_1 \dots f_n]$; // MSD
$Y \leftarrow [y_1 \dots y_m]$; // Target
$A \leftarrow [a_1 \dots a_m]$; // Salient alignments
Init: $X\_pos\_map = \{\}$, $F\_pos\_map = \{\}$ will store salient mappings from input positions to prediction steps, grouped by generation type.
for a_j in A:
  for P in a_j: ; // salient alignments to y_j
    if $P == X_i$: ; // aligned to lemma
      if $x_i == y_j$: ; // copy
        add j to $X\_pos\_map[X_i][c]$
      else:
        add j to $X\_pos\_map[X_i][g]$
    else ($P == F_k$): ; // aligned to tag
      add j to $F\_pos\_map[F_k]$
Outputs: X_pos_map, F_pos_map

Algorithm 5: (Step 3.1) Replace characters in Y with indexed generation type symbols using salient alignments to F

Inputs: F_pos_map; $\tilde{Y}$=copy(Y)
Init: $f2prev\_target = \{\}$; $f2ind = \{\}$
for F_k in F:
  if F_k in F_pos_map:
    for j in $F\_pos\_map[F_k]$: ; // Y indexes
      if f_k not in ($f2prev\_target$)):
        $f2ind[f_k]$=1; ; // If nothing was aligned yet to f_k, we create an index
        if $\tilde{y}_j$ is not replaced:
          $\tilde{y}_j \rightarrow f_k^1$
        else:
          $\tilde{y}_j$+= f_k^1
      else:
        if ($f2prev\_target[f_k] + 1$) != j: $f2ind[f_k]$+=1 ; // If something was aligned to f_k, check the last target step saved. Only increment it if it's not the same
        index=$f2ind[f_k]$
        if $\tilde{y}_j$ is not replaced:
          $\tilde{y}_j \rightarrow f_k^{index}$
        else:
          $\tilde{y}_j$+= f_k^{index}
Outputs: $P^{tr}(Y) = \tilde{Y}$

Algorithm 6: (Step 3.2) Replace characters in X and Y with indexed generation type symbols using salient alignments to X

Inputs: X_pos_map, $P^{tr}(Y)$
Init: $c_{index} = 1$; $g_{index} = 1$; $\tilde{X}$=copy(X); $\tilde{Y} = P^{dec}(Y)$
for X_i in X:
  if X_i in X_pos_map:
    $c(X_i) = X\_op\_map[X_i][c]$
    $g(X_i) = X\_op\_map[X_i][g]$
    if $c(X_i) == [j]$ and $x_i == y_j$ and $g(X_i) == \emptyset$: ; // X_i is 1-to-1 copy
      if X_{i-1} is not adjacent 1-to-1 copy:
        $c_{index} += 1$
      $\tilde{x}_i = C^{c_{index}}$; $\tilde{y}_j = C^{c_{index}}$
    else:
      if $c(X_i) != \emptyset$ and $g(X_i) != \emptyset$:
        $mask$ =‘’; $full\_index = []$
        for k in $c(X_i)$:
          c_{index}+=1
          add c_{index} to $full\_index$
          if $\tilde{y}_k$ is not replaced:
            $\tilde{y}_k \rightarrow C^{c\_index}$
          else:
            $\tilde{y}_k$+= C^{c_index}
        $mask$+=‘C^{full_index},
        $full\_index = []$
        for k in $g(X_i)$:
          g_{index}+=1
          add g_{index} to $full\_index$
          if $\tilde{y}_k$ is not replaced:
            $\tilde{y}_k \rightarrow G^{g\_index}$
          else:
            $\tilde{y}_k$+=G^{g_index}
        $mask$+=‘G^{full_index},
        $\tilde{x}_j \rightarrow mask$
Outputs: $P^{tr}(X) = \tilde{X}$, $P^{tr}(Y) = \tilde{Y}$

Appendix B

Self-Att^{sub} Lemma Patterns

Query	No. of/Acc	Patterns
gold target=*koot & MSD=msd_fin	37/0.97	*aa:8/1.0 (kar|sa |st |aa) *ua:5/1.0 (ku |or |ett |ua) *id*:5/1.0 (pro |mo |vo |id |a) *a:4/1.0 (pu |r |je |hti |a) *ta:4/1.0 (sk |r |uud |a |ta) *taa:4/1.0 (jo |kel |taa) *illa:2/1.0 (aal |to |illa) *ella:2/0.5 (n |ar |a |hd |ella) *ttaa:1/1.0 (ha |h |mo |ttaa) *sia:1/1.0 (har |sia), *ista:1/1.0 (li |i |pa |ista)
gold target=*kööt & MSD=msd_fin	9/1.00	*ä:3/1.0 (v |et |ele |hti |ä) *ää:3/1.0 (jä |n |ist |ää) *tä:2/1.0 (kä |pä |tä) *tää:1/1.0 (hy |mä |h |ää)

TABLE B.1: Finnish Self-Att^{sub} Patterns. MSD query *msd_fin* is *V;ACT;PRS;POS;IMP;3;PL*. Number of examples (*No of*) and accuracy (*Acc*) are shown per selection with query and per group pattern. For each query, we list all extracted lemma patterns (sorted by frequency in a decreasing order) along with one segmented lemma example (in parentheses) mapped to the pattern.

Query	No. of/Acc	Patterns
gold target=*scono & MSD=msd_it	23/1.00	*re: 9/1.0 (in| z| o| ti| chi| re) *ire: 7/1.0 (s| col| or| ire) *ir: 6/1.0 (re| in| ser| ir| si) *cir*: 1/1.0 (in| fer| o| cir |si)
gold target=*ano & MSD=msd_it	189/1.00	*are: 149/1.0 (z| am| pic| are) *arsi: 26//1.0 (im| pa| per| arsi) *car*:3/1.0 (ri|mb|ec|car|si) *izzarsi:2/1.0 (dest|abil|izz|arsi) *iarsi:2/1.0 (di|lan|i|arsi) *par*:2/1.0 (dis|col|par|si) *ciarsi:1/1.0 (au|to|den|un|ci|arsi) *mar*:1/1.0 (in|for|mar|si) *rarsi:1/1.0 (gi|ost|r|arsi) *itarsi:1/1.0 (ri|abil|it|arsi) *quar*:1/1.0 (sci|ac|quar|si)
gold target=*ono & !(*scono) & MSD=msd_it	41/0.95	*ere: 18/0.95 (ri| otten| ere) *dere: 10/1.0 (te| le| ve| dere) *ger*: 3/1.0 (cos| par| ger| si) *re:3/1.0 (servi|re) *e:1/0.0 (ri|ro|m|per|e) *ir*:1/1.0 (1908:s|ent|ir|si) *si:1/1.0 (es|p|or|si) *ire:1/1.0 (ri|di|ven|ire) *er*:1/1.0 (r|aggi|ung|er|si) *mer*:1/1.0 (ass|u|mer|si)

TABLE B.2: Italian Self-Att^{sub} Patterns. MSD query *msd_it* is *V;IND;PRS;3;PL*. Number of examples (*No of*) and accuracy (*Acc*) are shown per selection with query and per group pattern. For each query, we list all extracted lemma patterns (sorted by frequency in a decreasing order) along with one segmented lemma example (in parentheses) mapped to the pattern.

Bibliography

Ackerman, Farrell, James P. Blevins, and Robert Malouf (2009). "Parts and wholes: Patterns of relatedness in complex morphological systems and why they matter". In: *Analogy in grammar: Form and acquisition*, pp. 54–82.

Ackerman, Farrell and Robert Malouf (2013). "Morphological organization: The low conditional entropy conjecture". In: *Language*, pp. 429–464.

Ács, Judit (Oct. 2018). "BME-HAS System for CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection". In: *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*. Brussels: Association for Computational Linguistics, pp. 121–126. DOI: 10.18653/v1/K18-3016. URL: https://www.aclweb.org/anthology/K18-3016.

Aharoni, Roee and Yoav Goldberg (July 2017). "Morphological Inflection Generation with Hard Monotonic Attention". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 2004–2015. DOI: 10.18653/v1/P17-1183. URL: https://www.aclweb.org/anthology/P17-1183.

Allauzen, Cyril and Michael Riley (2012). "A Pushdown Transducer Extension for the OpenFst Library". In: *Implementation and Application of Automata - 17th International Conference, CIAA 2012, Porto, Portugal, July 17-20, 2012. Proceedings*, pp. 66–77.

Bahdanau, Dzmitry, Kyung Hyun Cho, and Yoshua Bengio (Jan. 2015). "Neural machine translation by jointly learning to align and translate". English (US). In: 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015.

Baron, Alistair and Paul Rayson (2008). "VARD 2: A tool for dealing with spelling variation in historical corpora". In: *Postgraduate Conference in Corpus Linguistics, Aston University*.

Belinkov, Yonatan, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass (Mar. 2020). "On the Linguistic Representational Power of Neural Machine Translation Models". In: *Computational Linguistics* 46.1, pp. 1–52. DOI: 10.1162/coli_a_00367. URL: https://www.aclweb.org/anthology/2020.cl-1.1.

Belinkov, Yonatan and James Glass (Mar. 2019). "Analysis Methods in Neural Language Processing: A Survey". In: *Transactions of the Association for Computational Linguistics* 7, pp. 49–72. DOI: 10.1162/tacl_a_00254. URL: https://www.aclweb.org/anthology/Q19-1004.

Bender, Emily M (2011). "On achieving and evaluating language-independence in NLP". In: *Linguistic Issues in Language Technology* 6.3, pp. 1–26.

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Janvin (Mar. 2003). "A Neural Probabilistic Language Model". In: *J. Mach. Learn. Res.* 3.null, 1137–1155. ISSN: 1532-4435.

Bentz, Christian, Tatyana Ruzsics, Alexander Koplenig, and Tanja Samardžić (Dec. 2016). "A Comparison Between Morphological Complexity Measures: Typological Data vs. Language Corpora". In: *Proceedings of the Workshop on Computational Linguistics for Linguistic Complexity (CL4LC)*. Osaka, Japan: The COLING 2016 Organizing Committee, pp. 142–153. URL: https://www.aclweb.org/anthology/W16-4117.

Bergmanis, Toms and Sharon Goldwater (Apr. 2017). "From Segmentation to Analyses: a Probabilistic Model for Unsupervised Morphology Induction". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 337–346. URL: https://www.aclweb.org/anthology/E17-1032.

Berko, Jean (1958). "The Child's Learning of English Morphology". In: *Word* 14.2-3, pp. 150–177.

Bickel, Balthasar and Johanna Nichols (2007). "Inflectional morphology". In: *Language typology and syntactic description* 3.2, pp. 169–240.

Bollmann, Marcel, Joachim Bingel, and Anders Søgaard (July 2017). "Learning attention for historical text normalization by learning to pronounce". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 332–344. DOI: 10.18653/v1/P17-1031. URL: https://www.aclweb.org/anthology/P17-1031.

Bollmann, Marcel, Stefanie Dipper, Julia Krasselt, and Florian Petran (2012). "Manual and semi-automatic normalization of historical spelling - case studies from Early New High German". In: *11th Conference on Natural Language Processing, KONVENS 2012, Empirical Methods in Natural Language Processing, Vienna, Austria, September 19-21, 2012*, pp. 342–350. URL: http://www.oegai.at/konvens2012/proceedings/51_bollmann12w/.

Bollmann, Marcel and Anders Søgaard (Dec. 2016). "Improving historical spelling normalization with bi-directional LSTMs and multi-task learning". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, pp. 131–139. URL: https://www.aclweb.org/anthology/C16-1013.

Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer (1993). "The Mathematics of Statistical Machine Translation: Parameter Estimation". In: *Computational Linguistics* 19.2, pp. 263–311. URL: https://www.aclweb.org/anthology/J93-2003.

Brunner, Gino, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer (2019). "On identifiability in transformers". In: *International Conference on Learning Representations*.

Bybee, Joan L (1985). *Morphology: A study of the relation between meaning and form*. Vol. 9. John Benjamins Publishing.

Chakrabarty, Abhisek, Onkar Arun Pandit, and Utpal Garain (July 2017). "Context Sensitive Lemmatization Using Two Successive Bidirectional Gated Recurrent Networks". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 1481–1491. DOI: 10.18653/v1/P17-1136. URL: https://www.aclweb.org/anthology/P17-1136.

Chen, Stanley F. and Joshua Goodman (June 1996). "An Empirical Study of Smoothing Techniques for Language Modeling". In: *34th Annual Meeting of the Association for Computational Linguistics*. Santa Cruz, California, USA: Association for Computational Linguistics, pp. 310–318. DOI: 10.3115/981863.981904. URL: https://www.aclweb.org/anthology/P96-1041.

Cho, Kyunghyun, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio (Oct. 2014a). "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches". In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association for Computational Linguistics, pp. 103–111. DOI: 10.3115/v1/W14-4012. URL: https://www.aclweb.org/anthology/W14-4012.

Cho, Kyunghyun, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (Oct. 2014b). "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. URL: https://www.aclweb.org/anthology/D14-1179.

Cotterell, Ryan, Christo Kirov, Mans Hulden, and Jason Eisner (Mar. 2019). "On the Complexity and Typology of Inflectional Morphological Systems". In: *Transactions of the Association for Computational Linguistics* 7, pp. 327–342. DOI: 10.1162/tacl_a_00271. URL: https://www.aclweb.org/anthology/Q19-1021.

Cotterell, Ryan, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Arya D. McCarthy, Katharina Kann, Sebastian Mielke, Garrett Nicolai, Miikka Silfverberg, David Yarowsky, Jason Eisner, and Mans Hulden (Oct. 2018). "The CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection". In: *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*. Brussels: Association for Computational Linguistics, pp. 1–27. DOI: 10.18653/v1/K18-3001. URL: https://www.aclweb.org/anthology/K18-3001.

Cotterell, Ryan, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden (Aug. 2017). "CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection in 52 Languages". In: *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*. Vancouver: Association for Computational Linguistics, pp. 1–30. DOI: 10.18653/v1/K17-2001. URL: https://www.aclweb.org/anthology/K17-2001.

Cotterell, Ryan, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden (Aug. 2016). "The SIGMORPHON 2016 Shared Task—Morphological Reinflection". In: *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Berlin, Germany: Association for Computational Linguistics, pp. 10–22. DOI: 10.18653/v1/W16-2002. URL: https://www.aclweb.org/anthology/W16-2002.

Cotterell, Ryan, Thomas Müller, Alexander Fraser, and Hinrich Schütze (July 2015). "Labeled Morphological Segmentation with Semi-Markov Models". In: *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Beijing, China: Association for Computational Linguistics, pp. 164–174. DOI: 10.18653/v1/K15-1017. URL: https://www.aclweb.org/anthology/K15-1017.

Cotterell, Ryan and Hinrich Schütze (2018). "Joint Semantic Synthesis and Morphological Analysis of the Derived Word". In: *Transactions of the Association for Computational Linguistics* 6, pp. 33–48. DOI: 10.1162/tacl_a_00003. URL: https://www.aclweb.org/anthology/Q18-1003.

Cotterell, Ryan, Tim Vieira, and Hinrich Schütze (June 2016). "A Joint Model of Orthography and Morphological Segmentation". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, pp. 664–669. DOI: 10.18653/v1/N16-1080. URL: https://www.aclweb.org/anthology/N16-1080.

Creutz, Mathias, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke (2007). "Morph-based Speech Recognition and Modeling of Out-of-vocabulary Words Across Languages". In: *ACM Trans. Speech Lang. Process.* 5.1, 3:1–3:29.

Creutz, Mathias and Krister Linden (2004). *Morpheme segmentation gold standards for Finnish and English*. Helsinki University of Technology.

Cybenko, G. (Dec. 1989). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals, and Systems (MCSS)* 2.4, pp. 303–314. ISSN: 0932-4194. DOI: 10.1007/BF02551274. URL: http://dx.doi.org/10.1007/BF02551274.

Dasgupta, Sajib and Vincent Ng (Apr. 2007). "High-Performance, Language-Independent Morphological Segmentation". In: *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. Rochester, New York: Association

for Computational Linguistics, pp. 155–163. URL: https://www.aclweb.org/anthology/N07-1020.

De Clercq, Orphée, Sarah Schulz, Bart Desmet, Els Lefever, and Véronique Hoste (Sept. 2013). "Normalization of Dutch User-Generated Content". In: *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*. Hissar, Bulgaria: INCOMA Ltd. Shoumen, BULGARIA, pp. 179–188. URL: https://www.aclweb.org/anthology/R13-1024.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (June 2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://www.aclweb.org/anthology/N19-1423.

Dieth, Eugen (1986). *Schwyzertütschi Dialäktschrift. Dieth-Schreibung*. ger. Ed. by Christian Schmid. 2. Aufl. Reihe lebendige Mundart ARRAY(0x563716230ee0). Übernommen aus FD-Bereich mit HST: Schwitzertütschi Dialäktschrift. Aarau [u.a.]: Sauerländer, 64 S. ISBN: 3-7941-2832-X and 978-3-7941-2832-7.

Dozat, Timothy, Peng Qi, and Christopher D. Manning (Aug. 2017). "Stanford's Graph-based Neural Dependency Parser at the CoNLL 2017 Shared Task". In: *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Vancouver, Canada: Association for Computational Linguistics, pp. 20–30. DOI: 10.18653/v1/K17-3002. URL: https://www.aclweb.org/anthology/K17-3002.

Dryer, Matthew S and Martin Haspelmath (2013a). "The world atlas of language structures online". In:

Dryer, Matthew S. and Martin Haspelmath, eds. (2013b). *WALS Online*. Leipzig: Max Planck Institute for Evolutionary Anthropology. URL: https://wals.info/.

Dyer, Christopher, Smaranda Muresan, and Philip Resnik (June 2008). "Generalizing Word Lattice Translation". In: *Proceedings of ACL-08: HLT*. Columbus, Ohio: Association for Computational Linguistics, pp. 1012–1020. URL: https://www.aclweb.org/anthology/P08-1115.

Edunov, Sergey, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato (June 2018). "Classical Structured Prediction Losses for Sequence to Sequence Learning". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 355–364. DOI: 10.18653/v1/N18-1033. URL: https://www.aclweb.org/anthology/N18-1033.

Eifring, H. and R. Theil (2005). *Linguistics for Students of Asian and African Languages*. Universitetet i Oslo: [available at http://www.uio.no/studier/emner/hf/ikos/EXFAC03-AAS/h05/larestoff/linguistics/].

Elman, Jeffrey L. (1990). "Finding structure in time". In: *Cognitive Science* 14.2, pp. 179–211.

Fraser, Alexander, Marion Weller, Aoife Cahill, and Fabienne Cap (Apr. 2012). "Modeling Inflection and Word-Formation in SMT". In: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Avignon, France: Association for Computational Linguistics, pp. 664–674. URL: https://www.aclweb.org/anthology/E12-1068.

Gage, Philip (1994). "A new algorithm for data compression". In: *The C Users Journal* 12.2, pp. 23–38.

Gesmundo, Andrea and Tanja Samardžić (July 2012). "Lemmatisation as a Tagging Task". In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Jeju Island, Korea: Association for Computational Linguistics, pp. 368–372. URL: https://www.aclweb.org/anthology/P12-2072.

Goldberg, Yoav (2017). *Neural Network Methods for Natural Language Processing*. Vol. 10. 1, pp. 1–309. DOI: 10.2200/S00762ED1V01Y201703HLT037. eprint: https://doi.org/10.2200/S00762ED1V01Y201703HLT037. URL: https://doi.org/10.2200/S00762ED1V01Y201703HLT037.

Goldwater, Sharon, Thomas L. Griffiths, and Mark Johnson (2006). "Interpolating between types and tokens by estimating power-law generators". In: *In Advances in Neural Information Processing Systems 18*, p. 18.

Goodman, Joshua T. (2001). "A bit of progress in language modeling". In: *Computer Speech & Language* 15.4, pp. 403 –434. ISSN: 0885-2308. DOI: https://doi.org/10.1006/csla.2001.0174. URL: http://www.sciencedirect.com/science/article/pii/S0885230801901743.

Graves, Alex (2008). "Supervised sequence labelling with recurrent neural networks." PhD thesis. Technical University Munich, pp. 1–117.

Gulcehre, Caglar, Orhan Firat, Kelvin Xu, Kyunghyun Cho, and Yoshua Bengio (Sept. 2017). "On Integrating a Language Model into Neural Machine Translation". In: *Comput. Speech Lang.* 45.C, 137–148. ISSN: 0885-2308. DOI: 10.1016/j.csl.2017.01.014. URL: https://doi.org/10.1016/j.csl.2017.01.014.

Haspelmath, Martin (2010). *Understanding Morphology*. London: Hodder Education.

Heafield, Kenneth (July 2011). "KenLM: Faster and Smaller Language Model Queries". In: *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Edinburgh, Scotland: Association for Computational Linguistics, pp. 187–197. URL: https://www.aclweb.org/anthology/W11-2123.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Honnet, Pierre-Edouard, Andrei Popescu-Belis, Claudiu Musat, and Michael Baeriswyl (May 2018). "Machine Translation of Low-Resource Spoken Dialects: Strategies for Normalizing Swiss German". In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*. Miyazaki, Japan:

European Languages Resources Association (ELRA). URL: https://www.aclweb.org/anthology/L18-1597.

Hornik, Kurt, Maxwell B. Stinchcombe, and Halbert White (1989). "Multilayer feedforward networks are universal approximators." In: *Neural Networks* 2.5, pp. 359–366. URL: http://dblp.uni-trier.de/db/journals/nn/nn2.html#HornikSW89.

Jacovi, Alon and Yoav Goldberg (2020). "Towards Faithfully Interpretable NLP Systems: How should we define and evaluate faithfulness?" In: *arXiv preprint arXiv:2004.03685*.

Jurafsky, Dan and James H Martin (2020). *Speech and Language Processing*. URL: https://web.stanford.edu/~jurafsky/slp3/.

Kanerva, Jenna, Filip Ginter, and Tapio Salakoski (2020). "Universal Lemmatizer: A sequence-to-sequence model for lemmatizing Universal Dependencies treebanks". In: *Natural Language Engineering*, 1–30. DOI: 10.1017/S1351324920000224.

Kann, Katharina, Ryan Cotterell, and Hinrich Schütze (Nov. 2016). "Neural Morphological Analysis: Encoding-Decoding Canonical Segments". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 961–967. DOI: 10.18653/v1/D16-1097. URL: https://www.aclweb.org/anthology/D16-1097.

Kann, Katharina and Hinrich Schütze (Aug. 2016). "Single-Model Encoder-Decoder with Explicit Morphological Representation for Reinflection". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 555–560. DOI: 10.18653/v1/P16-2090. URL: https://www.aclweb.org/anthology/P16-2090.

Keshava, Samarth and Emily Pitler (2006). "A simpler, intuitive approach to morpheme induction". In: *Proceedings of 2nd Pascal Challenges Workshop*, pp. 31–35.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1412.6980.

Kneser, Reinhard and Hermann Ney (1995). "Improved backing-off for M-gram language modeling." In: *ICASSP*. IEEE Computer Society, pp. 181–184. ISBN: 0-7803-2431-5. URL: http://dblp.uni-trier.de/db/conf/icassp/icassp1995.html#KneserN95.

Koehn, Philipp and Hieu Hoang (June 2007). "Factored Translation Models". In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: Association for Computational Linguistics, pp. 868–876. URL: https://www.aclweb.org/anthology/D07-1091.

Kondratyuk, Daniel, Tomáš Gavenčiak, Milan Straka, and Jan Hajič (2018). "LemmaTag: Jointly Tagging and Lemmatizing for Morphologically Rich Languages

with BRNNs". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 4921–4928. DOI: 10.18653/v1/D18-1532. URL: https://www.aclweb.org/anthology/D18-1532.

Korchagina, Natalia (May 2017). "Normalizing Medieval German Texts: from rules to deep learning". In: *Proceedings of the NoDaLiDa 2017 Workshop on Processing Historical Language*. Gothenburg: Linköping University Electronic Press, pp. 12–17. URL: https://www.aclweb.org/anthology/W17-0504.

Lee, Jason, Kyunghyun Cho, and Thomas Hofmann (2017). "Fully Character-Level Neural Machine Translation without Explicit Segmentation". In: *Transactions of the Association for Computational Linguistics* 5, pp. 365–378. DOI: 10.1162/tacl_a_00067. URL: https://www.aclweb.org/anthology/Q17-1026.

Ling, Wang, Chris Dyer, Alan W Black, Isabel Trancoso, Ramón Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís (Sept. 2015a). "Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1520–1530. DOI: 10.18653/v1/D15-1176. URL: https://www.aclweb.org/anthology/D15-1176.

Ling, Wang, Isabel Trancoso, Chris Dyer, and Alan W. Black (2015b). "Character-based Neural Machine Translation". In: *CoRR* abs/1511.04586. arXiv: 1511.04586. URL: http://arxiv.org/abs/1511.04586.

Lipton, Zachary C. (2018). "The mythos of model interpretability". In: *Commun. ACM* 61.10, pp. 36–43. DOI: 10.1145/3233231. URL: https://doi.org/10.1145/3233231.

Lison, Pierre, Jörg Tiedemann, and Milen Kouylekov (May 2018). "OpenSubtitles2018: Statistical Rescoring of Sentence Alignments in Large, Noisy Parallel Corpora". In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*. Miyazaki, Japan: European Languages Resources Association (ELRA). URL: https://www.aclweb.org/anthology/L18-1275.

Ljubesic, Nikola, Tomaz Erjavec, and Darja Fiser (2014). "Standardizing Tweets with Character-Level Machine Translation". In: *Computational Linguistics and Intelligent Text Processing - 15th International Conference, CICLing 2014, Kathmandu, Nepal, April 6-12, 2014, Proceedings, Part II*, pp. 164–175. DOI: 10.1007/978-3-642-54903-8_14. URL: https://doi.org/10.1007/978-3-642-54903-8_14.

Luong, Thang, Hieu Pham, and Christopher D. Manning (Sept. 2015). "Effective Approaches to Attention-based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1412–1421. DOI: 10.18653/v1/D15-1166. URL: https://www.aclweb.org/anthology/D15-1166.

Makarov, Peter, Tatiana Ruzsics, and Simon Clematide (Aug. 2017). "Align and Copy: UZH at SIGMORPHON 2017 Shared Task for Morphological Reinflection". In: *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*. Vancouver: Association for Computational Linguistics, pp. 49–57. DOI: 10.18653/v1/K17-2004. URL: https://www.aclweb.org/anthology/K17-2004.

Manning, D Christopher, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy (2020). "Emergent linguistic structure in artificial neural networks trained by self-supervision". In: *Proceedings of the National Academy of Sciences of the United States of America*.

Martinet, André (1967). "Eléments de linguistique générale". In: *Colin*.

Martins, Andre and Ramon Astudillo (2016). "From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification". In: ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, pp. 1614–1623. URL: http://proceedings.mlr.press/v48/martins16.html.

McMillan-Major, Angelina (Jan. 2020). "Automating Gloss Generation in Interlinear Glossed Text". In: *Proceedings of the Society for Computation in Linguistics 2020*. New York, New York: Association for Computational Linguistics, pp. 355–366. URL: https://www.aclweb.org/anthology/2020.scil-1.42.

Mikolov, T., S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur (2011). "Extensions of recurrent neural network language model". In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5528–5531. DOI: 10.1109/ICASSP.2011.5947611.

Naradowsky, Jason and Sharon Goldwater (2009). "Improving Morphology Induction by Learning Spelling Rules". In: *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pp. 1531–1536. URL: http://ijcai.org/Proceedings/09/Papers/256.pdf.

Narasimhan, Karthik, Regina Barzilay, and Tommi Jaakkola (2015). "An Unsupervised Method for Uncovering Morphological Chains". In: *Transactions of the Association for Computational Linguistics* 3, pp. 157–167. DOI: 10.1162/tacl_a_00130. URL: https://www.aclweb.org/anthology/Q15-1012.

Nicolai, Garrett, Colin Cherry, and Grzegorz Kondrak (2015). "Inflection Generation as Discriminative String Transduction". In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, pp. 922–931. DOI: 10.3115/v1/N15-1093. URL: https://www.aclweb.org/anthology/N15-1093.

Och, Franz Josef (July 2003). "Minimum Error Rate Training in Statistical Machine Translation". In: *Proceedings of the 41st Annual Meeting of the Association for*

Computational Linguistics. Sapporo, Japan: Association for Computational Linguistics, pp. 160–167. DOI: 10.3115/1075096.1075117. URL: https://www.aclweb.org/anthology/P03-1021.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2013). "On the difficulty of training recurrent neural networks." In: *ICML (3)*. Vol. 28. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 1310–1318. URL: http://dblp.uni-trier.de/db/conf/icml/icml2013.html#PascanuMB13.

Peters, Ben and André FT Martins (2019). "IT–IST at the SIGMORPHON 2019 Shared Task: Sparse Two-headed Models for Inflection". In: *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pp. 50–56.

Pettersson, Eva, Beáta Megyesi, and Joakim Nivre (May 2013). "Normalisation of Historical Text Using Context-Sensitive Weighted Levenshtein Distance and Compound Splitting". In: *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013)*. Oslo, Norway: Linköping University Electronic Press, Sweden, pp. 163–179. URL: https://www.aclweb.org/anthology/W13-5617.

— (Apr. 2014). "A Multilingual Evaluation of Three Spelling Normalisation Methods for Historical Text". In: *Proceedings of the 8th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*. Gothenburg, Sweden: Association for Computational Linguistics, pp. 32–41. DOI: 10.3115/v1/W14-0605. URL: https://www.aclweb.org/anthology/W14-0605.

Plank, Barbara, Anders Søgaard, and Yoav Goldberg (Aug. 2016). "Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 412–418. DOI: 10.18653/v1/P16-2067. URL: https://www.aclweb.org/anthology/P16-2067.

Ranzato, Marc'Aurelio, Sumit Chopra, Michael Auli, and Wojciech Zaremba (2016). "Sequence Level Training with Recurrent Neural Networks". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1511.06732.

Reffle, Ulrich (2011). "Efficiently generating correction suggestions for garbled tokens of historical language". In: *Natural Language Engineering* 17.2, pp. 265–282. DOI: 10.1017/S1351324911000039. URL: https://doi.org/10.1017/S1351324911000039.

Roark, Brian, Richard Sproat, Cyril Allauzen, Michael Riley, Jeffrey Sorensen, and Terry Tai (July 2012). "The OpenGrm open-source finite-state grammar software libraries". In: *Proceedings of the ACL 2012 System Demonstrations*. Jeju Island, Korea: Association for Computational Linguistics, pp. 61–66. URL: https://www.aclweb.org/anthology/P12-3011.

Robertson, Alexander and Sharon Goldwater (June 2018). "Evaluating Historical Text Normalization Systems: How Well Do They Generalize?" In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 720–725. DOI: 10.18653/v1/N18-2113. URL: https://www.aclweb.org/anthology/N18-2113.

Ruef, Beni and Simone Ueberwasser (2013). "The Taming of a Dialect: Interlinear Glossing of Swiss German Text Messages". In: *Non-standard Data Sources in Corpus-based Research*. Ed. by Marcos Zampieri and Sascha Diwersy. Aachen, Germany, pp. 61–68.

Ruokolainen, Teemu, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo (Aug. 2013). "Supervised Morphological Segmentation in a Low-Resource Learning Setting using Conditional Random Fields". In: *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 29–37. URL: https://www.aclweb.org/anthology/W13-3504.

Ruzsics, T., M. Lusetti, A. Göhring, T. Samardžić, and E. Stark (2019). "Neural text normalization with adapted decoding and POS features". In: *Natural Language Engineering* 25.5, 585–605. DOI: 10.1017/S1351324919000391.

Ruzsics, Tatyana and Tanja Samardžić (Aug. 2017). "Neural Sequence-to-sequence Learning of Internal Word Structure". In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada: Association for Computational Linguistics, pp. 184–194. DOI: 10.18653/v1/K17-1020. URL: https://www.aclweb.org/anthology/K17-1020.

Ruzsics, Tatyana, Olga Sozinova, Ximena Gutierrez-Vasques, and Tanja Samardžić (Apr. 2021). "Interpretability for Morphological Inflection: from Character-level Predictions to Subword-level Rules". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics.

Samardžić, Tanja, Yves Scherrer, and Elvira Glaser (2015). "Normalising orthographic and dialectal variants for the automatic processing of Swiss German". In: Proceedings of the 7th Language and Technology Conference.

Samardžić, Tanja, Yves Scherrer, and Elvira Glaser (2015). "Normalising orthographic and dialectal variants for the automatic processing of Swiss German". In: *Proceedings of The 4th Biennial Workshop on Less-Resourced Languages*. ELRA.

Samardzic, Tanja, Yves Scherrer, and Elvira Glaser (2016). "ArchiMob - A Corpus of Spoken Swiss German". In: ed. by N. Calzolari, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, and S. Odijk J. &. P. Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016). Paris, France: European Language Resources Association (ELRA).

Samardžić, Tanja, Robert Schikowski, and Sabine Stoll (July 2015). "Automatic interlinear glossing as two-level sequence classification". In: *Proceedings of the 9th SIGHUM Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*. Beijing, China: Association for Computational Linguistics, pp. 68–72. DOI: 10.18653/v1/W15-3710. URL: https://www.aclweb.org/anthology/W15-3710.

Schachter, Paul and Fé T. Otanes (1983). *Tagalog Reference Grammar*. reprinted edition. Berkeley: University of California Press.

Scherrer, Yves and Nikola Ljubešić (2016). "Automatic normalisation of the Swiss German ArchiMob corpus using character-level machine translation". In: ed. by S. Dipper. Proceedings of the 13th Conference on Natural Language Processing (KONVENS). Bochum.

Scherrer, Yves, Tanja Samardžić, and Elvira Glaser (2019). "Digitising Swiss German: how to process and study a polycentric spoken language". In: *Language Resources and Evaluation* 53.4, pp. 735–769.

Schmid, Helmut (1994). "Probabilistic Part-of-Speech Tagging Using Decision Trees". In: *International Conference on New Methods in Language Processing*. Manchester, UK, pp. 44–49.

Schrimpf, Martin, Idan Blank, Greta Tuckute, Carina Kauf, Eghbal A. Hosseini, Nancy Kanwisher, Joshua Tenenbaum, and Evelina Fedorenko (2020). "Artificial Neural Networks Accurately Predict Language Processing in the Brain". In: *bioRxiv*. DOI: 10.1101/2020.06.26.174482. eprint: https://www.biorxiv.org/content/early/2020/06/27/2020.06.26.174482.full.pdf. URL: https://www.biorxiv.org/content/early/2020/06/27/2020.06.26.174482.

Schuster, Mike and Kuldip K. Paliwal (1997). "Bidirectional recurrent neural networks." In: *IEEE Trans. Signal Process.* 45.11, pp. 2673–2681. URL: http://dblp.uni-trier.de/db/journals/tsp/tsp45.html#SchusterP97.

Sennrich, Rico (Apr. 2017). "How Grammatical is Character-level Neural Machine Translation? Assessing MT Quality with Contrastive Translation Pairs". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 376–382. URL: https://www.aclweb.org/anthology/E17-2060.

Sennrich, Rico and Barry Haddow (Aug. 2016). "Linguistic Input Features Improve Neural Machine Translation". In: *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*. Berlin, Germany: Association for Computational Linguistics, pp. 83–91. DOI: 10.18653/v1/W16-2209. URL: https://www.aclweb.org/anthology/W16-2209.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (Aug. 2016). "Neural Machine Translation of Rare Words with Subword Units". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 1715–

1725. DOI: 10.18653/v1/P16-1162. URL: https://www.aclweb.org/anthology/P16-1162.

Shannon, Claude E. and Warren Weaver (1949). *The Mathematical Theory of Communication*. Urbana, IL: University of Illinois Press. ISBN: 978-0-252-72548-7.

Shopen, Timothy (1985). *Language Typology and Syntactic Description: Volume 3*. Vol. 3. Cambridge University Press.

Stark, Elisabeth and Petra Meier (2017). "Argument Drop in Swiss WhatsApp Messages. A Pilot Study on French and (Swiss) German". In: *Zeitschrift für französische Sprache und Literatur* 127.3, pp. 224–252. ISSN: 0044-2747. URL: https://doi.org/10.5167/uzh-157322.

Stark, Elisabeth, Simone Ueberwasser, and Anne Göhring (2014). *Corpus "What's up, Switzerland?"* Tech. rep. Switzerland: University of Zurich. URL: www.whatsup-switzerland.ch.

Stark, Elisabeth, Simone Ueberwasser, and Beni Ruef (2009-2015). *Swiss SMS Corpus, University of Zurich*. https://sms.linguistik.uzh.ch.

Stoll, Sabine, Jekaterina Mazara, and Balthasar Bickel (2017). "The acquisition of polysynthetic verb forms in Chintang". In: *The Oxford Handbook of Polysynthesis*. Ed. by Michael Fortescue, Marianne Mithun, and Nicholas Evans. Oxford: Oxford University Press, pp. 495–514.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). "Sequence to Sequence Learning with Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger. Vol. 27. Curran Associates, Inc., pp. 3104–3112. URL: https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

Tan, Samson, Shafiq Joty, Lav Varshney, and Min-Yen Kan (Nov. 2020). "Mind Your Inflections! Improving NLP for Non-Standard Englishes with Base-Inflection Encoding". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, pp. 5647–5663. DOI: 10.18653/v1/2020.emnlp-main.455. URL: https://www.aclweb.org/anthology/2020.emnlp-main.455.

Tang, Gongbo, Fabienne Cap, Eva Pettersson, and Joakim Nivre (Aug. 2018). "An Evaluation of Neural Machine Translation Models on Historical Spelling Normalization". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, pp. 1320–1331. URL: https://www.aclweb.org/anthology/C18-1112.

Tiedemann, Jörg (2009). "Character-Based PSMT for Closely Related Languages". In: *Proceedings of the 13th Annual conference of the European Association for Machine Translation*. Barcelona, Spain: European Association for Machine Translation. URL: https://www.aclweb.org/anthology/2009.eamt-1.3.

Ueberwasser, Simone and Elisabeth Stark (2017). "What's up, Switzerland? A corpus-based research project in a multilingual country". In: *Linguistik Online* 84.5. ISSN:

1615-3014. URL: https://bop.unibe.ch/linguistik-online/article/view/3849.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., pp. 5998–6008. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Vilar, David, Jan-Thorsten Peter, and Hermann Ney (June 2007). "Can We Translate Letters?" In: *Proceedings of the Second Workshop on Statistical Machine Translation*. Prague, Czech Republic: Association for Computational Linguistics, pp. 33–39. URL: https://www.aclweb.org/anthology/W07-0705.

Vylomova, Ekaterina, Jennifer White, Elizabeth Salesky, Sabrina J Mielke, Shijie Wu, Edoardo Ponti, Rowan Hall Maudslay, Ran Zmigrod, Josef Valvoda, Svetlana Toldova, et al. (2020). "SIGMORPHON 2020 Shared Task 0: Typologically Diverse Morphological Inflection". In: *arXiv preprint arXiv:2006.11572*.

Weaver, Warren (1955). "Translation". In: *Machine Translation of Languages*. Ed. by William N. Locke and A. Donald Boothe. Reprinted from a memorandum written by Weaver in 1949. Cambridge, MA: MIT Press, pp. 15–23.

Wu, Shijie and Ryan Cotterell (July 2019). "Exact Hard Monotonic Attention for Character-Level Transduction". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 1530–1537. DOI: 10.18653/v1/P19-1148. URL: https://www.aclweb.org/anthology/P19-1148.

Yasunaga, Michihiro, Jungo Kasai, and Dragomir Radev (June 2018). "Robust Multilingual Part-of-Speech Tagging via Adversarial Training". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 976–986. DOI: 10.18653/v1/N18-1089. URL: https://www.aclweb.org/anthology/N18-1089.

Zeman, Daniel, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinková, Jan Hajič jr., Jaroslava Hlaváčová, Václava Kettnerová, Zdeňka Urešová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran

Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonça, Tatiana Lando, Rattima Nitisaroj, and Josie Li (Aug. 2017). "CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies". In: *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Vancouver, Canada: Association for Computational Linguistics, pp. 1–19. DOI: 10.18653/v1/K17-3001. URL: https://www.aclweb.org/anthology/K17-3001.

Zipf, George Kingsley (1935). *The Psychobiology of Language*. New York, NY, USA: Houghton-Mifflin.